UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

Escuela de Ingeniería Electrónica y Telecomunicaciones

"Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre"

Tesis previa a la obtención del
título de Ingeniero en Electrónica y Telecomunicaciones

Autores:

Saúl Genaro Ochoa Ochoa
Esteban Eduardo Velecela Gallegos

Director:

Dr. Luis Ismael Minchala Ávila

Cuenca - Ecuador
2016

Resumen

Palabras clave: HMI, Java, MATLAB, OPC, open-source, PLC, SCADA, TCP/IP.

La utilización de un sistema SCADA y comunicación OPC en un sistema manufacturero aumenta el rendimiento y competitividad de la industria. Su implementación dentro de la industria representa una importante inversión de dinero y la empresa se limita a las características que el proveedor dispone en cada paquete de software. Una solución alternativa planteada en este proyecto implica utilizar código libre para realizar un sistema SCADA open-source basado en comunicación TCP/IP. El sistema tiene una arquitectura cliente-servidor. El servidor implementado en Python, controla las variables de los PLCs y la base de datos. El cliente SCADA está desarrollado en Java. La correcta comunicación entre los niveles de automatización del modelo CIM permite gestionar la planificación, control de producción y administración de recursos. Por esta razón, se desarrolla en este trabajo un cliente en MATLAB, que simula las funciones de un HMI, demostrando la interoperabilidad del sistema.

Abstract

Keywords: CIM, Java, open software, OSACIM, Python, SCADA.

The computer integrated manufacturing (CIM) approach allows the possibility to remotely and optimally control the entire production process within a plant. The implementation of CIM architectures demands the installation of several software platforms, which most of the cases are commercial and have high licensing prices. This research presents the development of open software for advanced CIM (OSACIM) by using two open software development platforms: Java and Python. The use of open software in the development of the solution allows the creation of a low price CIM approach. The results of the use of OSCACIM in laboratory tests show good results in comparison with a commercial SCADA system that performs the same features of the proposed software. A detailed methodology of the software development is presented, as well as the main features and limitations of the system.

Índice general

Índice de figuras

Índice de tablas

Yo, *Saúl Genaro Ochoa Ochoa*, autor de la tesis *"Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre"*, certifico que todas las ideas, opiniones, y contenidos expuestos en la presente investigación, son de exclusiva responsabilidad de sus autores.

Cuenca, Marzo 2016.

Saúl Genaro Ochoa Ochoa
C.I. 0104610662

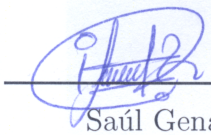
Yo, *Esteban Eduardo Velecela Gallegos*, autor de la tesis *"Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre"*, certifico que todas las ideas, opiniones, y contenidos expuestos en la presente investigación, son de exclusiva responsabilidad de sus autores.

Cuenca, Marzo 2016.

Esteban Eduardo Velecela Gallegos
C.I. 0302623772

Yo, *Saúl Genaro Ochoa Ochoa*, autor de la tesis *"Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre"*, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *Ingeniero Electrónico y en Telecomunicaciones*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, Marzo 2016.



Saúl Genaro Ochoa Ochoa
C.I. 0104610662

Yo, *Esteban Eduardo Velecela Gallegos*, autor de la tesis *"Propuesta e implementación de una solución alternativa para el control, supervisión y comunicación en procesos industriales mediante programas de código libre"*, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de *Ingeniero Electrónico y en Telecomunicaciones*. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, Marzo 2016.

Esteban Eduardo Velecela Gallegos
C.I. 0302623772

Agradecimientos

Agradecemos en primer lugar a Dios por brindarnos salud y vida. También, extender nuestra gratitud a las siguientes personas:

A todas las personas que nos ayudaron y apoyaron para culminar con éxito esta tesis.
A nuestros amigos y familiares por el apoyo incondicional durante la etapa de formación profesional.
Al Sr. Francisco Sánchez, encargado del laboratorio de máquinas, por permitirnos utilizar el laboratorio y por brindarnos su ayuda.
Y finalmente a nuestro director de tesis, Dr. Ismael Minchala por su valioso tiempo invertido en la dirección de esta tesis.

Dedicatoria

Dedico este trabajo a mi familia, en especial a mis padres: Eduardo e Irene que con sus consejos y apoyo me ayudar a superar mis metas planteadas. A mis hermanos: Santiago, Stalin, Lissethe y a mi hija Daniela quienes son el motor de mi vida y me impulsan a seguir superándome en mi vida profesional. Finalmente a Dios, por todos los favores recibidos durante el transcurso de mi carrera.

Esteban.

Dedico este trabajo a mi familia. A mi madre, Sara por ser mi inspiración y apoyo incondicional para alcanzar mis objetivos. A mis hermanos Wilmer y Klever por el apoyo brindado en las diferentes etapas de mi vida. A mis tíos: Ninfa y Jacinto por ser un ejemplo a seguir y brindarme sus buenos consejos.

Saúl.

Convenciones

Este trabajo utilizará las siguientes convenciones:

Tipo de letra	Decripción
Bold	Para definir palabras importantes.
Italic	Para definir palabras en inglés.
`Courier`	Para definir una clase.

Abreviaciones

API	Application Programming Interface
CIM	Computer Integrated Manufacturing
COM	Component Object Model
CP	Communication Processor
DHCP	Dynamic Host Configuration Protocol
GPL	General Public License
HMI	Human-Machine Interface
HOQ	House Of Quality
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LAN	Local Area Network
MIS	Management Information System
MTU	Master Central Unit
OLE	Object Linking and Embedding
OPC	Object Linking and Embedding for Process Control
OSACIM	Open Software for Advanced Computer Integrated Manufacturing
PDU	Power Distribution Unit
PLC	Programmable Logic Controller
QFD	Quality function Deployment
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
WAN	Wide Area Network
XML	Xtensible Markup Language

Capítulo 1

Introducción

1.1. Introducción

El manejo adecuado de la información, permite a las empresas tener un enfoque claro de sus operaciones. En las industrias se presentan problemas para resolver estos desafíos, y esto se refleja en niveles bajos de rendimiento, producción y competitividad dentro del mercado.

La implementación de sistemas de manufactura integrada por computador (CIM, por sus siglas en inglés) brinda una comunicación óptima entre los elementos fundamentales de la empresa. Esto se logra mediante la utilización de redes de computadores, para así formar un sistema altamente interconectado.

En los años noventa empezó el desarrollo de aplicativos que brindan información de las variables involucradas en los procesos industriales, para así mejorar las actividades de producción [1]. Un aplicativo que satisface dichos requerimientos es un sistema de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés), ya que permite la comunicación con los dispositivos de campo para controlar los procesos de forma automática.

Los altos costos de licenciamiento de sistemas SCADA han dificultado su penetración en el mercado local. Adicionalmente, limita los tipos o modelos de controladores y dispositivos de monitoreo [2]. Por tanto, este trabajo de inves-

tigación/innovación propone el desarrollo de un software libre para aplicaciones avanzada de manufactura integrada por computadora (OSACIM, por sus siglas en inglés).

1.2. Descripción del problema

El control y comunicación entre procesos en una industria juega un papel importante al realizar un análisis en su productividad. Los dispositivos utilizados en el control y comunicación son proporcionados por proveedores externos a las empresas. Por lo general estos dispositivos dependen de aplicativos con códigos fuente propietario, limitando a la industria a adquirir dispositivos de una determinada marca [3].

Un aplicativo que satisfaga estos requerimientos tendrá como elementos un sistema SCADA y un protocolo de comunicación industrial (OPC, por sus siglas en inglés), los cuales permitirán establecer una comunicación con dispositivos de campo, como los controladores lógicos programables (PLC, por sus siglas en inglés).

Una solución al problema planteado se basa en programas de código propietario. En [4] presentan como solución desarrollar un servidor OPC en LabVIEW 6i de National Instruments, para interconectar dispositivos de automatización. En [5] utilizan el software SCADA WinCC de Siemens, para implementar en un prototipo de empacadora de galletas.

El costo que representa implementar un sistema SCADA con comunicación OPC es una limitación para las industrias manufactureras. El desarrollo de un OSACIM, resuelve esta limitación.

1.3. Estado del Arte

La incorporación de nuevas tecnologías en sistemas manufactureros y sistemas eléctricos conllevan a la necesitan de establecer una comunicación en red eficiente entre todos los procesos [6]. El uso de sistemas SCADA permite una comunicación distribuida entre todos los componentes de la planta y así conocer el estado actual del sistema.

La incorporación de un sistema SCADA basado en código abierto permite a todas las empresas disponer de estas ventajas y así competir en igualdad de recursos con otras empresas. El desarrollo de sistemas SCADA a partir de hardware y software libre se ha incrementado en los últimos años.

En [7] se presenta el desarrollo de un sistema SCADA *open-source*. Los niveles de campo y control se basan en hardware *copyleft*. Un ejemplo de una plataforma *copyleft* es el hardware/software libre *ZotePLC*. Para el nivel de supervisión se utiliza el software HMI de código abierto *Proview*. Este software, desarrollado en Suecia, es un sistema de control orientado a objetos, programado en Linux, cuyas principales funciones son:

- Brindar un control de datos a tiempo real.
- Permite modificar y adquirir datos.
- Brinda almacenamiento y visualización de tendencias de los datos.

En [3] la arquitectura utilizada para el sistema SCADA *open-source* se basa en un ordenador central (RTU, por sus siglas en inglés) que se comunica con un Arduino. Los valores de los dispositivos de campo son obtenidos mediante una red de comunicación industrial Modbus [3], [8]. El funcionamiento del sistema es el siguiente:

- Se utiliza en software “Snap4Arduino”, su función es diseñar la interfaz gráfica de la unidad central maestra (MTU, por sus siglas en inglés).

- El protocolo “Standard Firmata” realiza la comunicación entre la RTU y el controlador.
- Se utiliza la plataforma “Arduino Mega” como controlador.
- Se conectan diferentes sensores en el controlador.

En [9] se propone un sistema SCADA *open-source* para un sistema de generación distribuida. La comunicación es inalámbrica y utiliza tecnología ZigBee, el cual se utiliza para intercambiar información entre el sistema de generación y el sistema SCADA. El sistema SCADA se desarrolló en Python, Java y Javascript. La base de datos utilizada es Derby.

La tendencia de los sistemas SCADA está orientada hacia servidores web y sistemas multiplataforma [10]. La necesidad de acceder desde una aplicación móvil a los datos de un sistema manufacturero es una de las exigencias del mercado actual.

En [11] se muestra el desarrollo de un aplicativo web SCADA, el cual es un software SCADA para clientes web. Este servicio se ofrece utilizando el protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés). Los clientes responden estas solicitudes mediante lenguaje de marcas extensible (XML, por sus siglas en inglés).

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una solución alternativa para el control, supervisión y comunicación en procesos industriales utilizando programas de código libre.

1.4.2. Objetivos específicos

1. Simular procesos industriales a través de hardware conectado en las entradas y salidas de los PLCs S7-1200.

2. Realizar un estudio teórico del estándar de comunicación OPC y sistema SCADA.

3. Desarrollar software para el control y supervisión, así como para la comunicación de los procesos industriales mediante código libre.

4. Comprobar la interoperabilidad del sistema utilizando un software con licenciamiento comercial.

1.5. Contribuciones de la tesis

Las contribuciones de la tesis son las siguientes:

- Reducción de problemas de comunicación entre dispositivos de automatización (PLCs, drivers, etc) a través de una alternativa de servidor de comunicación industrial de código abierto.
- Reducción importante de costos en la implementación de sistemas flexibles de manufactura integrada por computadora.
- Monitoreo, almacenamiento y control supervisorio a través de un software SCADA de código abierto.
- Recolección de diferentes tipos de datos y visualización de sus estados de acuerdo a los requerimientos de los usuarios del sistema.
- Redacción de un artículo técnico.

Capítulo 2

Marco teórico

2.1. Modelo de manufactura integrada por computador

El modelo de manufactura integrada por computador (CIM, por sus siglas en inglés) es un sistema estandarizado, de múltiples capas que integra todos los elementos fundamentales de la empresa. Con este modelo es fácil conocer el nivel de comunicación que se emplea dentro de un ambiente manufacturero.

Figura 2.1: Modelo teórico CIM

La Figura 2.1 muestra que existe una jerarquía para la comunicación en un sistema manufacturero.

El modelo CIM garantiza una mejor administración para los recursos de la empresa, combinando todas las unidades funcionales de la compañía, por medio de redes de computadores. Con esta estructura se logra un sistema altamente interconectado y con una respuesta rápida a las demandas del mercado [12]. La problemática en CIM se presenta al momento de integrar los niveles de este modelo. Incluso cuando los componentes operan correctamente en su nivel, las dificultades surgen cuando se desea interactuar entre niveles.

2.1.1. Nivel de campo

Es el nivel inferior del modelo CIM. La Figura 2.2 muestra que el nivel de campo está formado por sensores y actuadores, los cuales se encuentran distribuidos en la línea de producción.

Figura 2.2: Sensores y actuadores en una industria

En [13] se indican los aspectos más importantes a tener en cuenta en este nivel. Éstos son:

- Descripción y propiedades detalladas de los procesos (tipo de variable, rangos).
- Determinación de los lazos de control del proceso.
- Implementación de respaldos mecánicos para la instrumentación electrónica que controla variables importantes, para así garantizar el funcionamiento correcto de la planta en caso de un fallo en el sistema eléctrico.

2.1.2. Nivel de control

Los principales objetivos de este nivel son el monitoreo y la sincronización de los procesos básicos realizados en el nivel inferior. La Figura 2.3 muestra los componentes que suelen encontrarse en el nivel de control, los cuales son uno o varios controladores lógicos programables (PLCs), RTUs, entre otros.

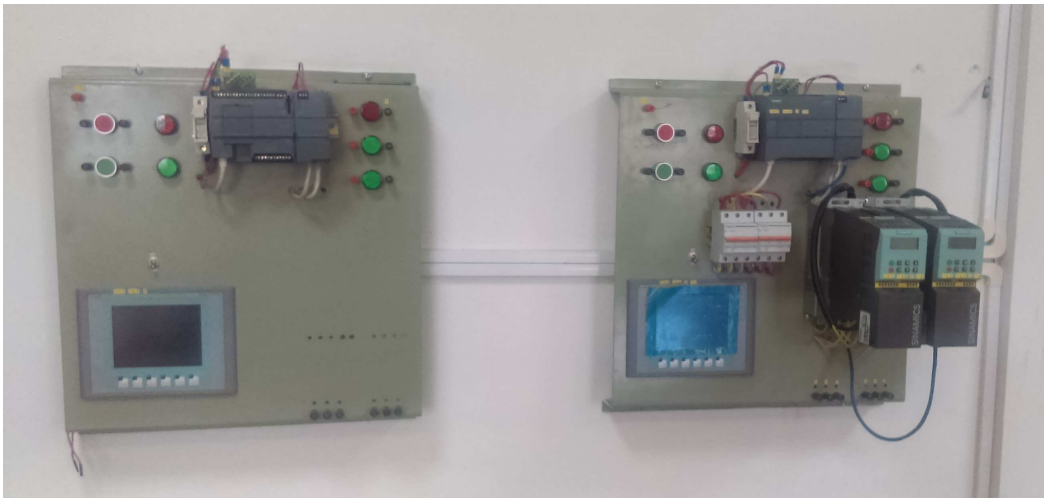


Figura 2.3: Nivel de control

Este conjunto de componentes conforma lo que se conoce como una célula de fabricación [14]. La información recibida en este nivel es proporcionada de manera permanente al nivel de supervisión, para que en el nivel 3 del modelo CIM sean programadas las acciones respectivas de cada proceso dentro de la planta.

2.1.3. Nivel de supervisión

En este nivel se ejecuta la adquisición de datos en tiempo real de la información originada en el nivel de control [15]. Como resultado se logra el sincronismo de las operaciones entre varias celdas de fabricación.

Las funciones principales de este nivel son las siguientes:

- La supervisión completa de los procesos.
- Monitoreo de las operaciones.
- La revisión y reparación de errores para garantizar el funcionamiento y fiabilidad de los procesos.
- El almacenamiento de la información de los procesos.

2.1.4. Nivel de planificación

Las debilidades y fortalezas de los procesos involucrados en la planta son identificadas en este nivel. Esto se logra manteniendo la comunicación con el nivel de supervisión utilizando los medios físicos de red, basándose en la información que tenemos a disposición.

Las principales tareas desarrolladas en este nivel son los siguientes:

- Planificación y control de producción.
- Diseño y definición de los procesos de fabricación y sus etapas.
- Administración de los recursos.
- Mantenimiento y gestión de calidad de los recursos.

2.1.5. Nivel de gestión

La administración corporativa se realiza en este nivel. Esto se logra al comunicar los niveles de planificación de una o varias plantas, obteniendo la información

en tiempo real de las variables involucradas en los procesos de producción [15].

Las características de este nivel son las siguientes:

- Administrar la producción completa de la empresa.
- Comunicación entre plantas.
- Mantener la relación cliente-servidor.
- Gestión de recursos humanos, tecnología y sistemas de información(MIS, por sus siglas en inglés).
- En este nivel se realiza la gestión y la integración de los niveles más bajos del modelo CIM.

2.2. Protocolo object linking and embedding (OLE) for process control (OPC)

2.2.1. Descripción

En la última década los sistemas de automatización han presentado un problema de comunicación, entre dispositivos o aplicaciones, debido al uso de protocolos propietarios. OPC presenta una alternativa de solución a este problema de interconectividad. La Figura 2.4 muestra que sin OPC cada aplicación requiere un controlador o driver para comunicarse con un equipo específico, esto limita considerablemente la interoperabilidad de equipos de diferentes fabricantes.

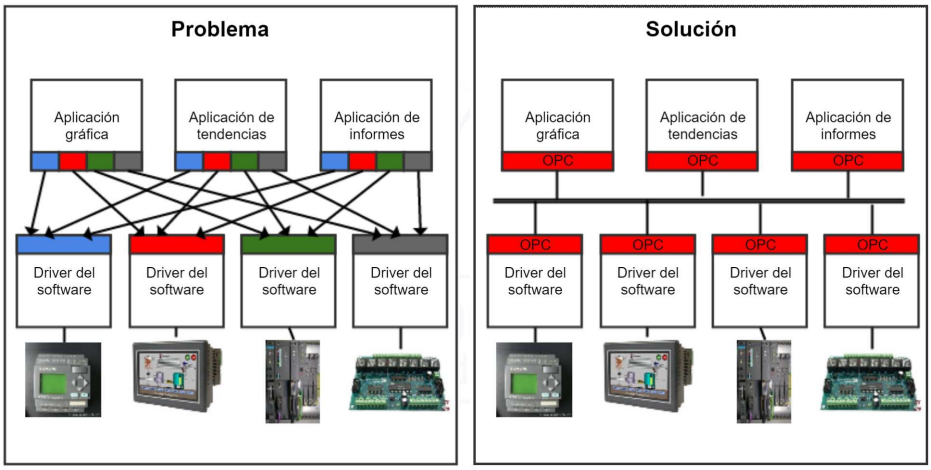
Como solución a este conflicto cinco empresas Microsoft, Opto-22, Rockwell Software e Intuitiv Software, Fisher-Rosemount, Intellution, definieron el protocolo OPC.

Los pilares fundamentales de OPC se detallan a continuación:

- Incrustación y enlazado de objetos (OLE, por siglas en inglés) se implementó como solución para el intercambio de información entre aplicaciones

de MS Windows. La principal ventaja del protocolo es su adecuado funcionamiento en sistemas de tiempo real, la cual es muy importante para los sistemas de automatización [16]. Funciona mediante el manejo de documentos compuestos a partir de los datos de dos aplicaciones diferentes, es decir un archivo es el principal y otro el contenedor. Cada cambio en el archivo principal se refleja en el documento contenedor [2].

- Modelo de objetos y componentes (COM, por sus siglas en inglés) es una plataforma dedicada a la apertura de componentes, ya que estos pueden ser utilizados en diferentes lenguajes de programación.



(a) Problema sin OPC (b) Solución OPC

Figura 2.4: Solución del problema con OPC [13]

OPC es un estándar de comunicación industrial, cuyo objetivo es intercambiar datos entre clientes, independiente de la tecnología utilizada para cumplir este objetivo. De esta manera OPC garantiza la comunicación con cualquier equipo que cumpla el estándar OPC, ya que cada driver se tendrá que utilizar solo una vez [17].

2.2.2. Arquitectura

El accionar de este protocolo está basado en una arquitectura con dos componentes fundamentales: cliente OPC y servidor OPC, los cuales intercambian

datos. Un sistema puede poseer múltiples clientes y servidores, por lo tanto una aplicación puede tener componentes ya sea del tipo cliente o del tipo servidor para así garantizar la interacción con otros elementos del sistema [18].

La Figura 2.5 muestra un ejemplo básico de una arquitectura OPC en donde se pueden apreciar tres componentes: el primero es del tipo cliente, el segundo de tipo servidor y un tercero que es la combinación del tipo cliente y servidor.

Figura 2.5: Arquitectura OPC de un sistema

Cliente OPC

La Figura 2.6 muestra los elementos típicos de un cliente OPC, como se relacionan entre sí y con los demás componentes del sistema.

La aplicación cliente es el código que implementa la función del cliente. Utiliza la interfaz de programación de aplicaciones (API, por sus siglas en inglés) de cliente OPC para enviar y recibir las solicitudes y respuestas (Servicio OPC) al Servidor OPC [19].

Un cliente OPC es un programa que asiste otra aplicación, con la función de obtener información de un servidor OPC y proveer a la aplicación, (por ejemplo un sistema SCADA).

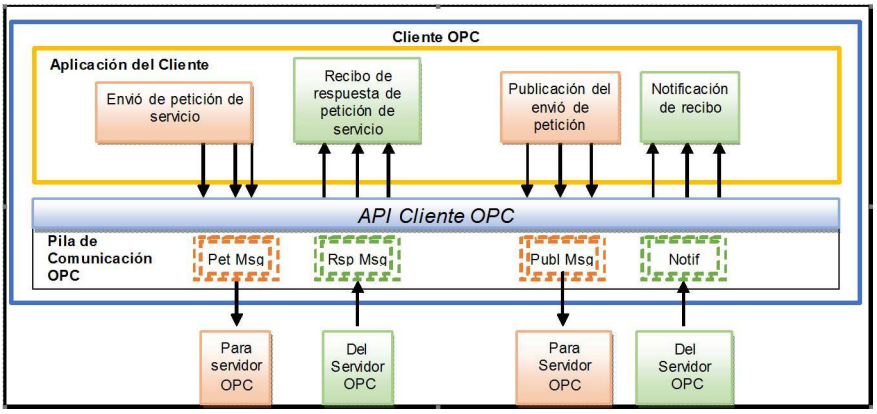


Figura 2.6: Arquitectura de un cliente OPC

Servidor OPC

Un servidor OPC es un aplicativo que cumple con una o más especificaciones del estándar OPC, generando un controlador estándar. La Figura 2.7 ilustra los elementos principales que conforman un servidor OPC y como se relacionan entre si.

Las principales características de un servidor OPC son:

- Un servidor OPC brinda interfaces OPC para las aplicaciones cliente.
- Un servidor debe ser capaz de manipular los datos en una fuente de información y proporcionarlos ante una petición de un cliente OPC.

2.2.3. Características de OPC

Los beneficios más importantes que se obtienen al implementar un servidor OPC son los siguientes:

- Un cliente OPC tiene la capacidad de comunicarse con cualquier servidor OPC, que se encuentre en la misma red, sin requerir un driver específico.
- Los fabricantes de software no deberán modificar sus drivers por cambios en las propiedades de su hardware.

- El usuario tiene la libertad de escoger los dispositivos o aplicaciones que satisfagan sus necesidades sin preocuparse por el fabricante que los realice.
- Los tipos de datos que soporta OPC son: datos históricos, datos de tiempo real, eventos y alarmas. Cada categoría soporta un gran número de tipos de variables como enteros, flotantes, cadenas de caracteres, fechas, etc.
- Un servidor OPC trabaja paralelamente con aplicaciones comerciales conocidas como Excel, Access, Power Builder, Visual Basic, etc [17].

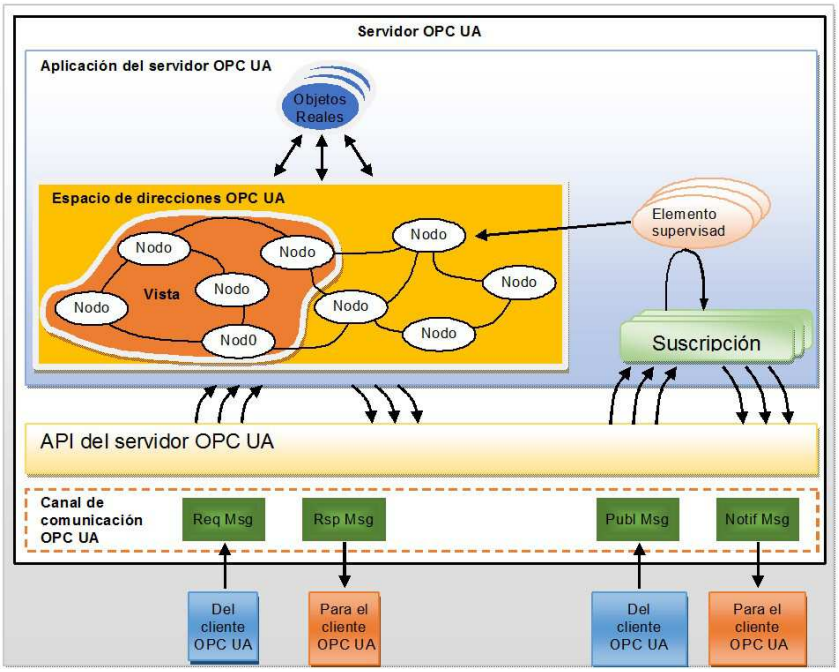


Figura 2.7: Arquitectura de un servidor OPC [19]

2.2.4. Funcionamiento de OPC

La arquitectura cliente-servidor implantada en OPC promueve el intercambio de información entre estos componentes. El principal actor en esta arquitectura es el servidor, puesto que es el que tiene acceso a la fuente de información (PLCs), poniendo a disposición las aplicaciones a los clientes.

Un servidor OPC puede brindar servicio a varios clientes. Uno de los aspectos fundamentales que debe desarrollar un servidor es la capacidad de conectividad y velocidad de respuesta a las peticiones de un cliente.

La Figura 2.8 muestra una topología de red en donde todos los elementos comparten el medio físico, los clientes pueden hacer peticiones a todos los servidores existentes, dependiendo de los dispositivos a controlar.

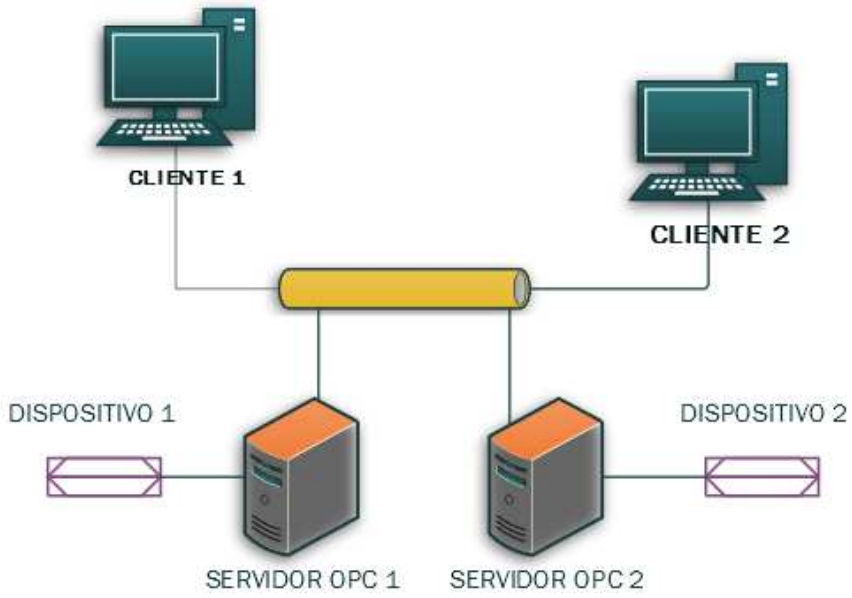


Figura 2.8: Clientes y servidores OPC compartiendo el medio de acceso físico

Mecanismo de funcionamiento

La implementación de una aplicación OPC determina los dispositivos o los datos que el servidor tiene acceso y control, los nombres de los datos, y detalles técnicos del dispositivo. Esta información es utilizada para generar una clase u objeto en el servidor.

Las clases u objetos del servidor son proporcionadas al cliente cuando este hace una petición de servicio, generando gran funcionalidad y flexibilidad al sistema.

Estructura de datos

El servidor actúa como contenedor de objetos y grupos. Los grupos contienen información de ellos y la organización lógica de los ítems. Los ítems representan información acerca de las conexiones a las fuentes de datos. La estructura de datos mencionada se ilustra en la Figura 2.7.

2.2.5. Tipos de servidores OPC

El estándar OPC incluye algunas especificaciones, que se distinguen por los datos y tipos de acceso. La *OPC foundation* ha definido cuatro tipos de servidores OPC.

OPC DA (Data Access) Server:
Proporciona un intercambio de datos en tiempo real. Los clientes y servidores son capaces de leer y escribir datos de las variables de los dispositivos involucrados en la planta [20].

OPC HDA (Historical Data Access) Server:
Permite el acceso a los datos que han sido almacenados. En un sistema SCADA, es decir un cliente OPC, los datos históricos pueden ser proporcionados por el servidor de una manera uniforme.

OPC A&E (Alarms and Events) Server:
Brinda notificaciones de eventos y alarmas a clientes OPC. Estos incluyen alarmas de proceso, acciones realizadas por el usuario, mensajes de información y seguimiento.

OPC UA (Unified Architecture) Server:
Es una combinación de todas las funciones especificadas en OPC (OPC DA, OPC HDA, OPC A& E). Sus características lo hacen escalable, seguro y además tiene alto rendimiento ya que trabaja con cualquier tipo de dato [21].

2.3. Sistemas de supervisión, control y adquisición de datos (SCADA)

2.3.1. Descripción

El sistema SCADA es un programa que tiene como finalidad el control y la supervisión de un sistema [17]. La información y comunicación de los dispositivos de campo son proporcionados al usuario por esta aplicación.

Un sistema SCADA se debe proyectar como una utilidad de supervisión y mando. Debe tener los siguientes objetivos:

- Perfecta funcionalidad de manejo y visualización en determinado sistema operativo.
- La arquitectura del sistema debe ser abierta, permitiendo crear soluciones de mando y visualización mediante programas estándares o de usuario.
- El sistema debe crecer de acuerdo a las necesidades del medio.
- Garantizar la seguridad e integridad de la información en cuanto al manejo y acceso.
- Posibilidad de programación numérica.

2.3.2. Arquitectura

La Figura 2.9 muestra tres grupos principales de un sistema SCADA, éstos son:

- Programa de adquisición de datos y control.
- Sistemas de adquisición y mando (sensores y actuadores).
- Sistema de interconexión (comunicaciones).

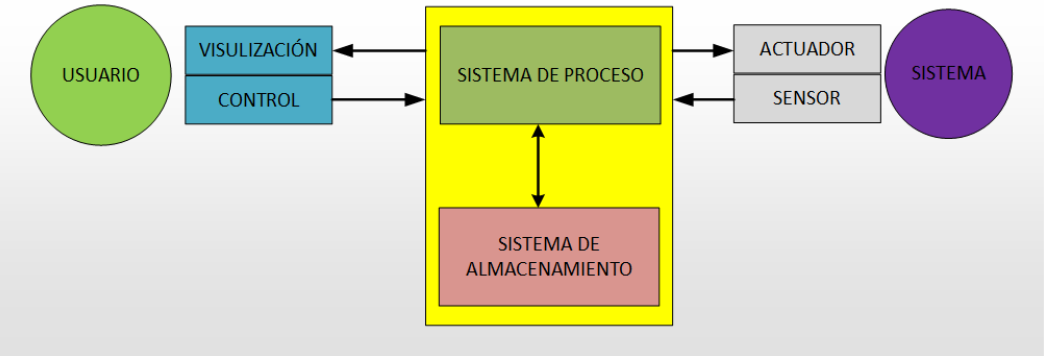


Figura 2.9: Arquitectura básica de un sistema SCADA [17]

La evolución de los sistemas SCADA se ha desarrollado conforme a los avances de las tecnologías de la computación moderna. En esta sección trataremos la arquitectura de las tres generaciones de los sistemas SCADA:

1. **Primera Generación:** Sistemas SCADA centralizados.
2. **Segunda Generación:** Sistemas SCADA distribuidos.
3. **Tercera Generación:** Sistemas SCADA en red.

Sistemas SCADA centralizados

Los primeros sistemas SCADA desarrollados con el concepto de computación en general se centraron en los sistemas de *mainframes*. Las redes de computadores eran inexistentes y cada SCADA era un solo sistema centralizado. El resultado era un sistema SCADA independiente con ninguna conectividad a otros sistemas.

Las redes de área amplia (WAN, por sus en inglés) se implementaron con un solo propósito para comunicarse con RTUs. Los protocolos WAN en ese momento eran desconocidos. Los protocolos de comunicación para redes SCADA fueron desarrollados por proveedores de equipos RTU y eran de uso propietario.

La redundancia en estos sistemas se logró mediante el uso de dos sistemas *mainframe* idénticamente equipadas, una primaria y una copia de seguridad, conectados

en el nivel de bus. La principal función del sistema de reserva era vigilar la primaria y asumir el control en caso de algún fallo. Este tipo de operación en espera significaba que poco o ningún procesamiento se realiza en el sistema de espera. La Figura 2.10 muestra la arquitectura SCADA típica de la primera generación [22].

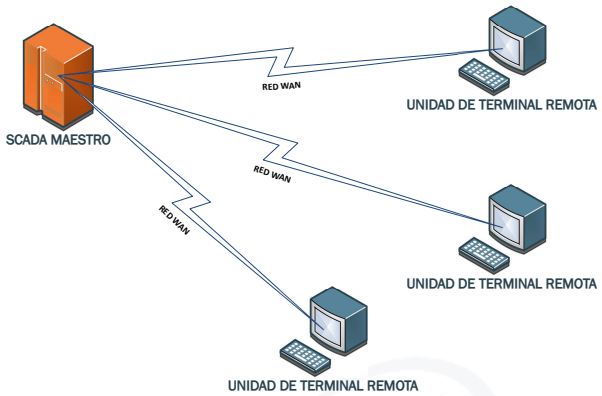


Figura 2.10: Arquitectura SCADA típica de la primera generación

Sistemas SCADA distribuidos

La siguiente generación de sistemas SCADA se aprovechó de la evolución y las mejoras en la miniaturización del sistema y la tecnología de red de área local (LAN, por sus siglas en inglés) para distribuir el procesamiento a través de múltiples sistemas. Múltiples estaciones, cada uno con una función específica, estaban conectados a una LAN compartiendo información en tiempo real [22].

Estas estaciones eran típicamente de la clase mini-ordenador, más pequeños y menos costosos que los procesadores de primera generación. La Figura 2.11 muestra la arquitectura de esta generación.

Sistemas SCADA en red

Esta arquitectura esta estrechamente relacionada con la segunda generación. La principal diferencia radica en el uso de sistemas abiertos en lugar de sistemas propietarios. La mejora en la tercera generación es la apertura de la arquitectura del sistema, utilizando estándares y protocolos abiertos, haciendo posible distribuir funcionalidades SCADA a través de una WAN y no sólo una LAN [22].

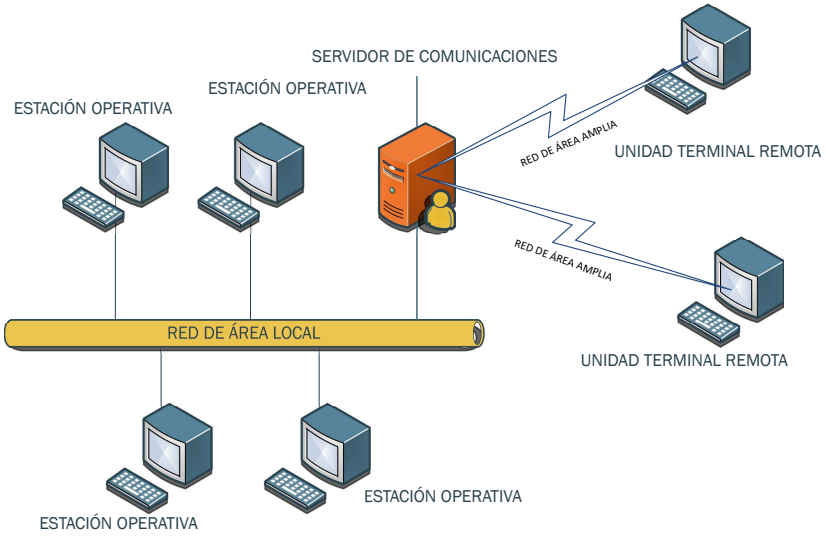


Figura 2.11: Arquitectura SCADA de la segunda generación [22]

Los estándares abiertos eliminan algunas de las limitaciones de las generaciones anteriores. La Figura 2.12 muestra la arquitectura de un sistema SCADA en red.

2.3.3. Criterios de diseño de un sistema de SCADA

La necesidad de un SCADA se refleja en el control de un proceso específico dentro de la industria. Este proceso debe cumplir ciertos requerimientos para implementar un sistema de control. Los requerimientos necesarios son:

- La cantidad de variables que dispone el proceso es alta.
- Los componentes del proceso están distribuidos.
- Las variables del proceso son requeridas a tiempo real.

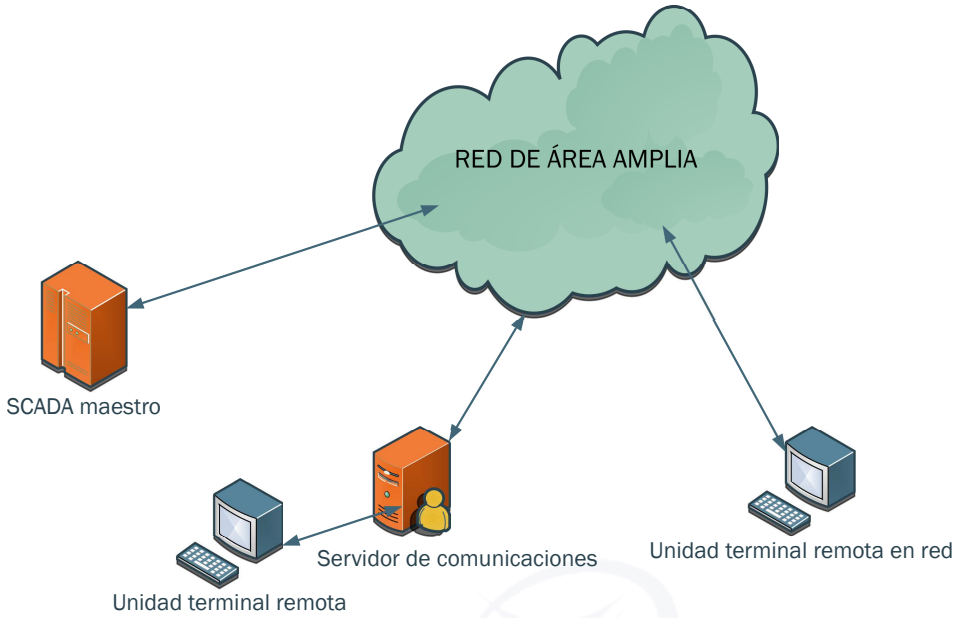


Figura 2.12: Arquitectura SCADA de la tercera generación [22]

Al definir la necesidad de un sistema SCADA, en cualquier sistema de control es indispensable el funcionamiento correcto del mismo durante todo el tiempo en el que los procesos monitoreados estén en ejecución. Los criterios de diseño para un sistema SCADA generalmente son [23]:

- **Disponibilidad:** Esto se refiere al proceso de control sin fallos, es decir la capacidad de acceder al sistema y que éste responda de acuerdo a las especificaciones de diseño.
- **Robustez:** Este criterio indica la eficiencia del sistema ante cualquier eventualidad interna o externa que se presente.
- **Seguridad:** Persigue la confiabilidad e integridad de los datos. Además la autenticación de los usuarios, para evitar un acceso a la información no deseada.
- **Prestaciones:** Hace referencia al tiempo de respuesta que tiene el sistema [23]. El tiempo de respuesta del sistema debe ser corto ya que las variables de proceso se requieren constantemente.

- **Mantenibilidad:** Es la probabilidad que tiene un sistema de operar en condiciones normales dentro de un tiempo establecido, luego de que haya ocurrido un fallo [24]. Para lograr esto, se requiere un sistema con bajo tiempo de mantenimiento.
- **Escalabilidad:** Es la capacidad de crecimiento del sistema.

2.3.4. Características de un sistema SCADA

Las principales características de un sistema SCADA son las siguientes [25]:

- Almacenamiento, supervisión, adquisición y control de los datos de manera fluida y confiable.
- Interfaz gráfica, preferentemente animada, de los procesos controlados. El estado de las variables del proceso se presentan al operador mediante alarmas.
- Compatibilidad con diferentes aplicaciones y bases de datos. Los datos adquiridos por el sistema SCADA pueden representarse de manera gráfica.
- Transmisión de datos descentralizado con PLCs y otros PC.
- Información detallada al operador del funcionamiento de las variables de control, tanto las variables que no funcionan normalmente (alarmas) como el cambio de estado o valor que se produce en las variables de la planta (eventos) [25].
- Permite la comunicación entre un elemento determinado de campo y un sistema de control implementado en una PC.
- Seguridad de acceso a los datos mediante autenticación.

Capítulo 3

Desarrollo del Software de supervisión y control

3.1. Configuración de la red

El OSACIM esta implementado sobre el sistema de pre molienda de una industria de cemento. Para representar las variables más importantes se ha utilizado 2 PLCs, que representan mediante indicadores los procesos que tiene esta área. Los pasos para la configuración de los PLCs y la interfaz humano-máquina (HMI, por sus siglas en inglés) se presentan en las siguientes subsecciones.

3.1.1. Programación en Step 7 TIA PORTAL V12

La ventana principal de Step 7 se muestra en la Figura 3.1. Los pasos que se requieren para iniciar un proyecto son:

- **Crear proyecto:** Esta opción crea un nuevo proyecto, definiendo su nombre y la ubicación en la que se guardará.
- **Agregar dispositivo:** La Figura 3.2 muestra los dispositivos de la familia Siemens que se pueden utilizar dentro del proyecto. Los dispositivos que se agregarán al proyecto son el PLC SIEMENS S7-1200 y el HMI. Para seleccionar un PLC se debe conocer el tipo de PLC y la versión de firmware que permita cargar correctamente el dispositivo.

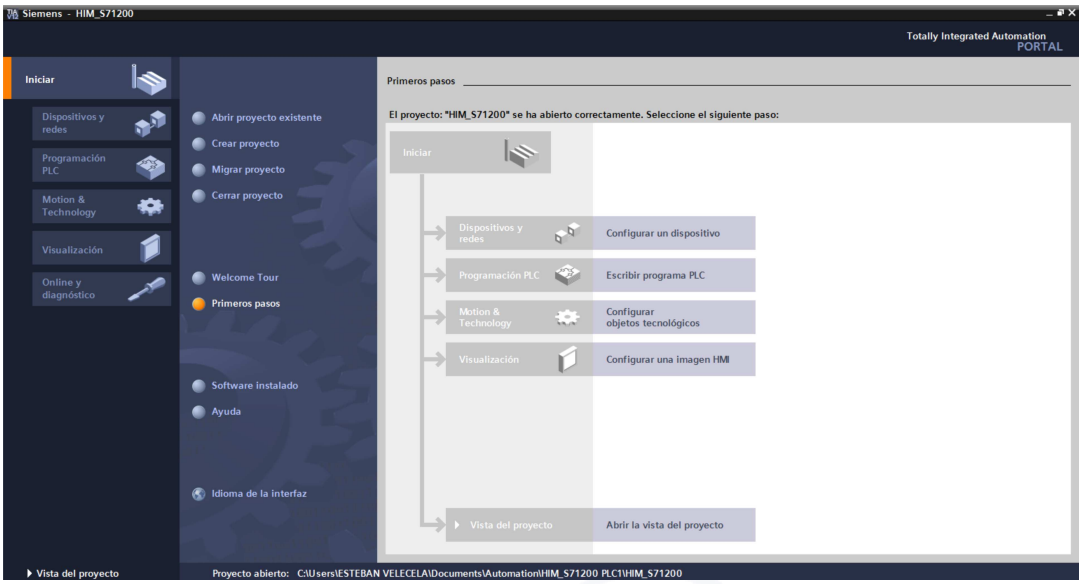


Figura 3.1: Interfaz principal del programa TIA PORTAL v12

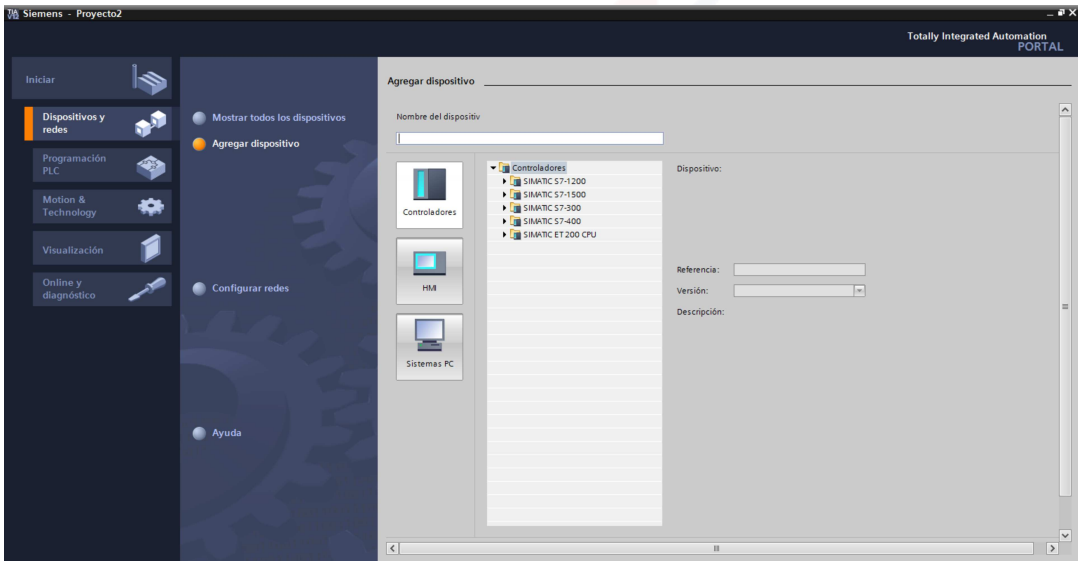


Figura 3.2: Opciones para agregar dispositivos en el programa TIA PORTAL v12

Luego de crear el proyecto, se configura la dirección de Protocolo de Internet (IP,por sus siglas en inglés) del dispositivo dentro de la propiedad “interfaz PROFINET” del PLC. Esto se muestra en la Figura 3.3.

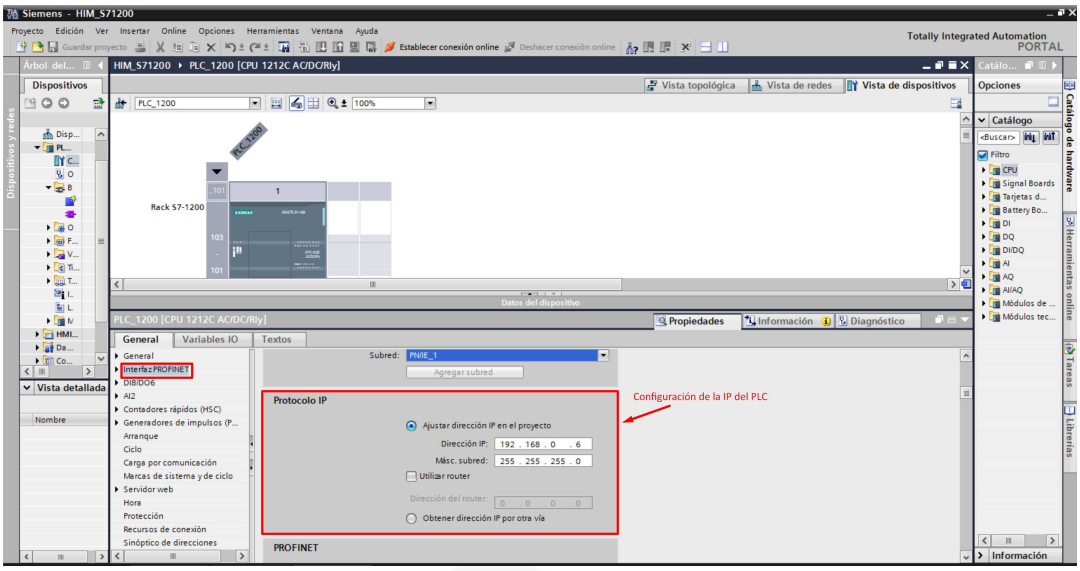


Figura 3.3: Configuración de la dirección IP del PLC en Step 7

Las variables implementadas en el PLC son agregadas como se muestra en la Figura 3.4. Éstas son ingresadas al programa mediante el nombre, tipo y dirección lógica.

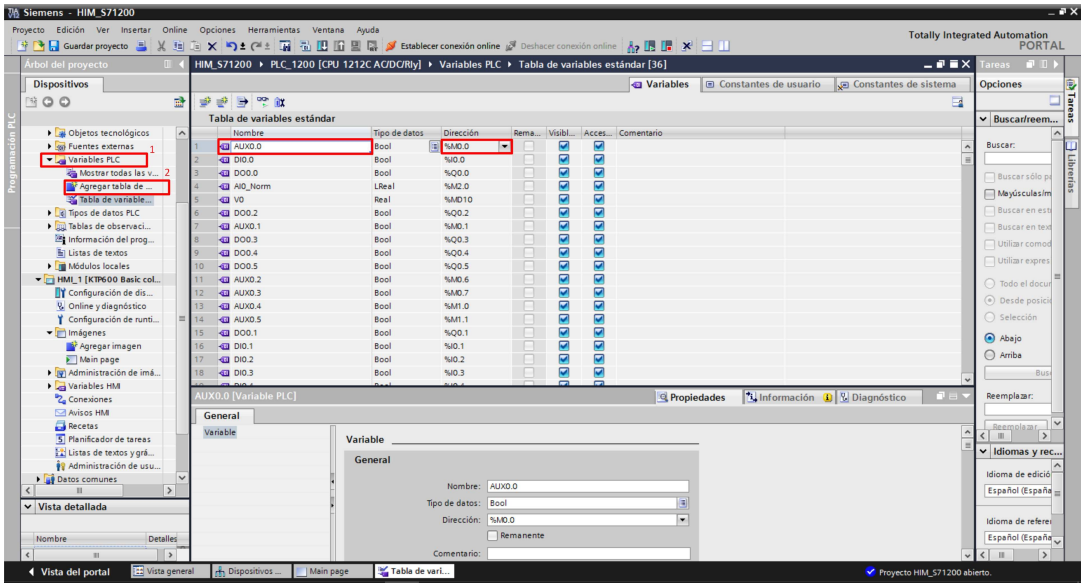


Figura 3.4: Variables utilizadas en el PLC

Los datos ingresados son utilizados y referenciados de acuerdo al tipo de programación que se utilice dentro del Step 7. Para programar el PLC existen 3 tipos de lenguaje de programación, éstos son:

- **FUP:** Su programación es gráfica, utiliza diagrama de funciones.
- **KOP:** Su programación es gráfica, utiliza esquema de contactos.
- **AWL:** Su programación es orientada a la máquina mediante lista de instrucciones.

La Figura 3.5 muestra el lenguaje de programación KOP utilizado dentro del proyecto. Cada contactor está asignado a una variable auxiliar. Esta variable controla el estado, mediante la función asignación, de su salida digital.

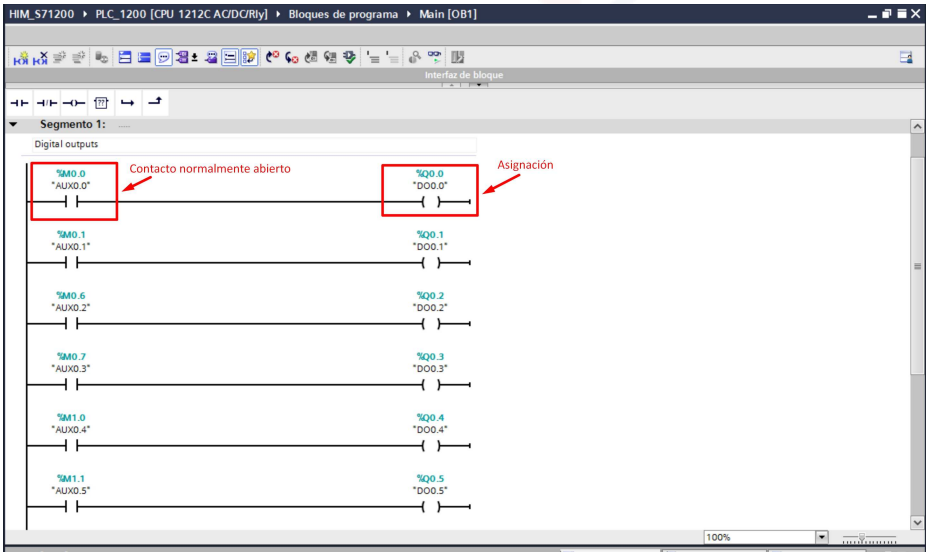


Figura 3.5: Esquema de contactores dentro de Step7

Con la configuración y programación del PLC lista, dentro del proyecto se agrega una pantalla HMI KTP-600 Basic color PN. La Figura 3.6 muestra los pasos que se debe seguir para agregar un HMI al proyecto.

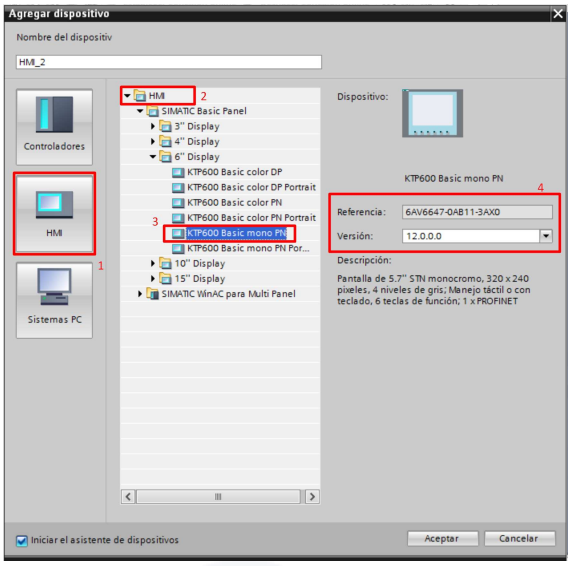


Figura 3.6: Pasos para agregar un HMI en Step 7

La configuración de la dirección IP del HMI se muestra en la Figura 3.7. Su dirección IP y máscara de red se configuran de acuerdo al dispositivo utilizado.

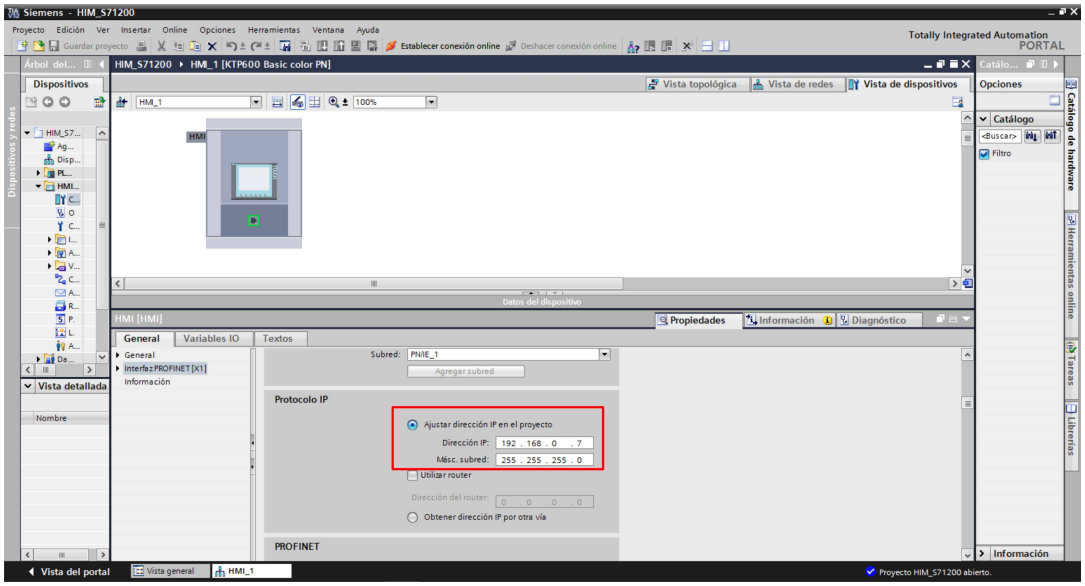


Figura 3.7: Configuración de la dirección IP del HMI en Step 7

El HMI puede ser programado desde el software Step 7. La Figura 3.8 muestra las herramientas que pueden utilizarse dentro de la interfaz gráfica.

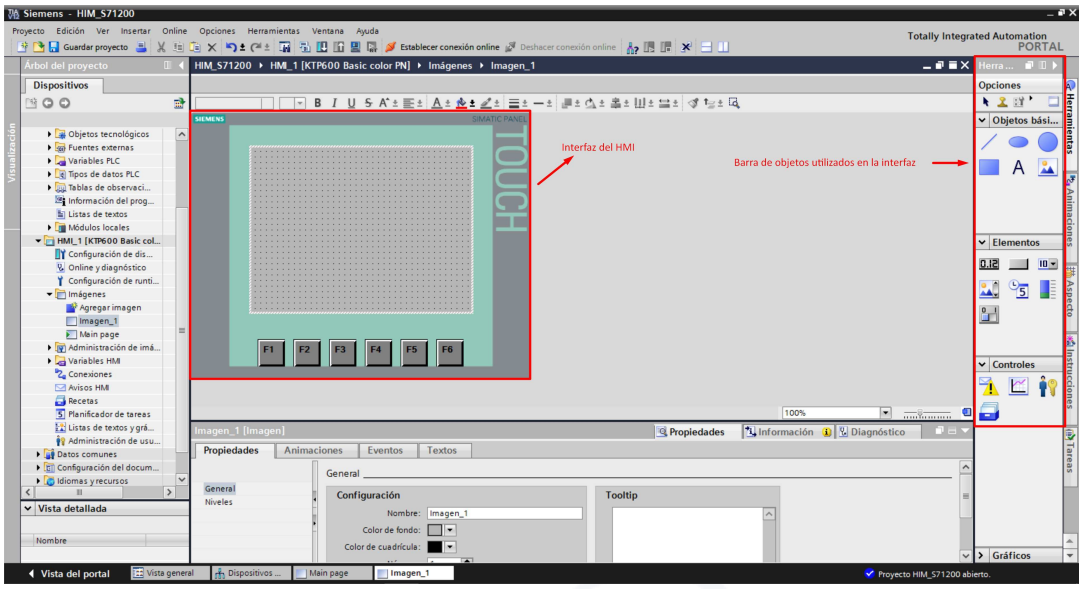


Figura 3.8: Herramientas disponibles para el HMI dentro del software Step 7

Estas herramientas permiten crear: botones, indicadores digitales e indicadores analógicos, etc, como se muestra en la Figura 3.9. Cada objeto ingresado en la interfaz HMI tiene que ser asignado a una variable del PLC y así exista una interacción entre estos dos dispositivos.

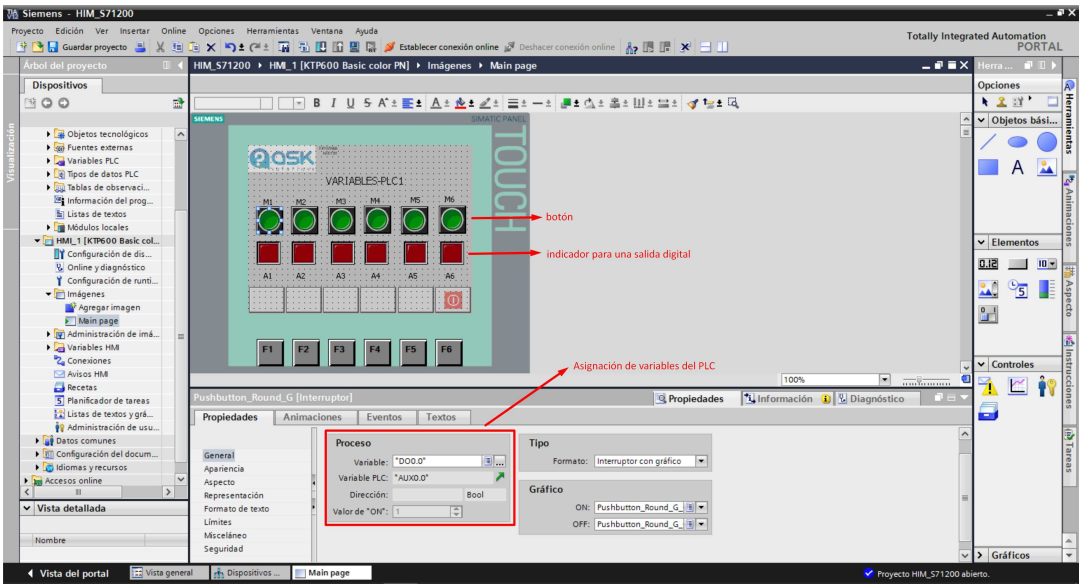
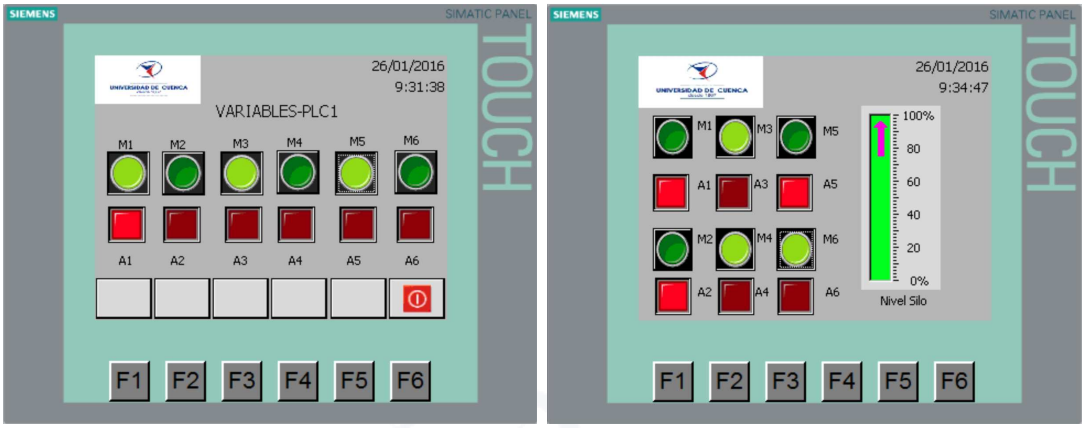


Figura 3.9: Interfaz HMI dentro del software Step 7

Finalmente, cada programa se carga en el PLC y en el HMI. La Figura 3.10 muestra la interfaz que se utiliza para el monitoreo de las variables de cada PLC. Los botones e indicadores cumplen funciones específicas.



(a) Interfaz HMI para el PLC1 (b) Interfaz HMI para el PLC2

Figura 3.10: Interfaces finales implementadas en la red

La Tabla 3.1 muestra la variable que controla cada botón e indicador del HMI para el PLC1. La Tabla 3.2 muestra la variable que controla cada botón e indicador del HMI para el PLC2.

Tabla 3.1: Dirección lógica de cada indicador del HMI para el PLC1

Nombre	Dirección	Tipo
M1	M0.0	booleano
M2	M0.1	booleano
M3	M0.2	booleano
M4	M0.6	booleano
M5	M1.0	booleano
M6	M1.1	booleano
A1	I0.0	booleano
A2	I0.1	booleano
A3	I0.2	booleano
A4	I0.3	booleano
A5	I0.4	booleano
A6	I0.5	booleano

Tabla 3.2: Dirección lógica de cada indicador del HMI para el PLC2

Nombre	Dirección	Tipo
M1	M0.0	booleano
M2	M0.1	booleano
M3	M0.2	booleano
M4	M0.3	booleano
M5	M0.4	booleano
M6	M0.5	booleano
A1	I0.0	booleano
A2	I0.1	booleano
A3	I0.2	booleano
A4	I0.3	booleano
A5	I0.4	booleano
A6	I0.5	booleano
Nivel Silo	IW64	entero

3.2. Desarrollo de la comunicación

La comunicación del OSACIM trabaja sobre una red de área local Ethernet. Los equipos existentes en el laboratorio de la universidad están configurados en red, con la dirección IP 192.168.0.0 y máscara tipo B. La Figura 3.11 muestra el diagrama de red del sistema.

La Tabla 3.3 muestra el direccionamiento de la red. La red implementada trabaja sobre una sola dirección IP. Es posible omitir la configuración de una puerta de enlace. Las configuraciones de las computadoras donde se ejecutan las aplicaciones de cliente y servidor se hacen manualmente imponiendo una dirección IP física estática puesto que no disponemos de un servidor de protocolo de configuración dinámica de host (DHCP, por sus siglas en inglés).

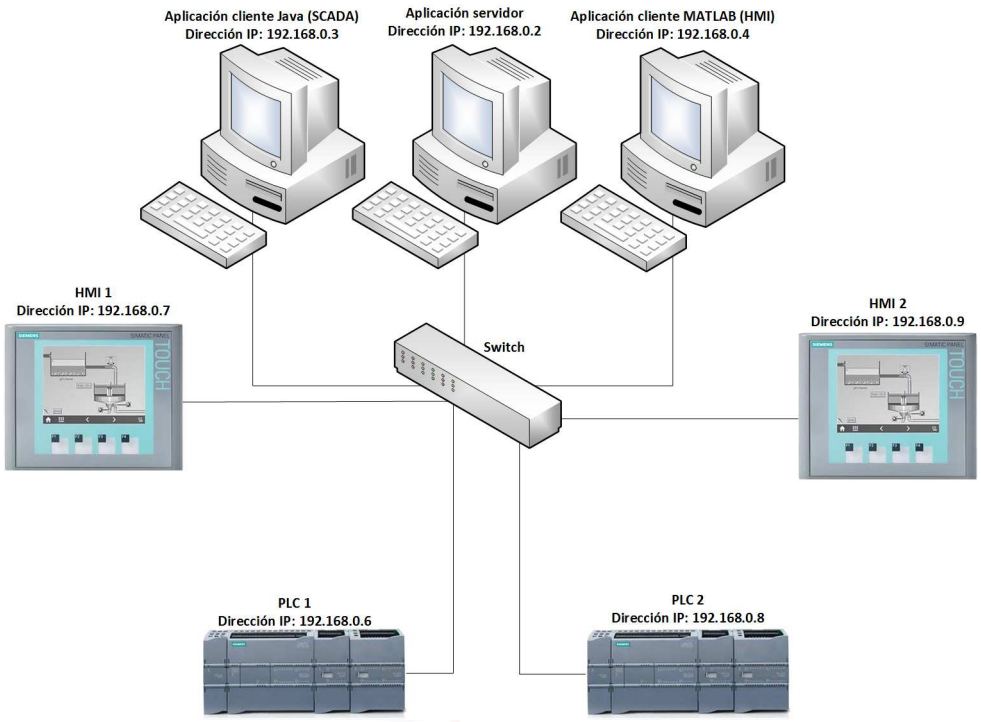


Figura 3.11: Diagrama de red

Tabla 3.3: Tabla de direccionamiento de la red

Dispositivo	**Dirección IP**	**Máscara de subred**
Aplicación servidor	192.168.0.2	255.255.255.0
Aplicación cliente SCADA	192.168.0.3	255.255.255.0
Aplicación cliente HMI	192.168.0.4	255.255.255.0
HMI 1 (Siemens)	192.168.0.7	255.255.255.0
HMI 2 (Siemens)	192.168.0.9	255.255.255.0
PLC 1(S7-1200)	192.168.0.6	255.255.255.0
PLC 2 (S7-1200)	192.168.0.6	255.255.255.0

3.3. Desarrollo del sistema

3.3.1. Descripción del sistema planteado

Existen diferentes procesos en la fabricación de cemento de la industria Guapán. Todos estos procesos se dividen en áreas, siendo el área G una de las más importantes ya que realiza los procesos de: secado de la puzolana, premolienda y el molino de cemento [26].

El sistema SCADA desarrollado simula las componentes principales del proceso de premolienda ubicada en el área G. La Figura 3.12 muestra el esquema de flujo para el proceso de premolienda. Este sistema fue implementado en 2 PLCs, debido al número de variables requeridas.

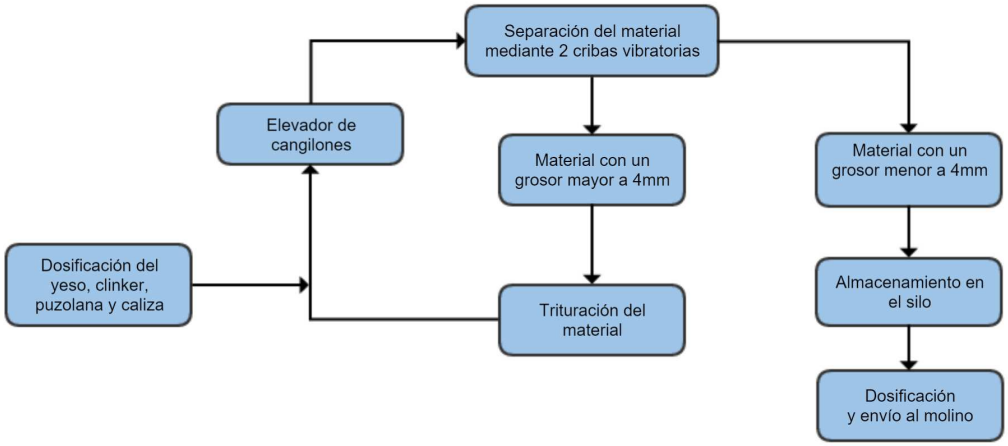


Figura 3.12: Esquema de flujo del proceso de premolienda

El PLC1 implementa los siguientes procesos:

- La dosificación de la caliza, clinker, yeso y puzolana.
- El transporte del material hacia el elevador de cangilones.

El PLC2 implementa los siguientes procesos:

- Separación del material dependiendo de su grosor.

- Trituración del material grueso, si su diámetro es mayor a 4mm.
- Almacenamiento del material en un silo para luego ser enviado hacia el molino.

El funcionamiento correcto de estos procesos se indica mediante alarmas. Las alarmas implementadas para el PLC1 son:

- Desvío del flujo de la puzolana.
- Desvío del flujo del clinker.
- Desvío del flujo del yeso.
- Desvío del flujo del aditivo.
- Desvío del flujo total de almacenamiento.
- Falla en la banda del elevador de cangilones.

Para el PLC2, las alarmas implementadas son:

- Alta vibración en la criba vibratoria.
- Temperatura del separador 1.
- Temperatura del separador 2.
- Falla en el sistema de trituración.
- Desvío de la banda transportadora 1.
- Desvío de la banda transportadora 2.
- Nivel del silo de almacenamiento.

3.3.2. Desarrollo del servidor

Comunicación con los dispositivos de campo

La librería que permite la comunicación entre los dispositivos de campo y el servidor es Snap7. Snap7 utiliza el protocolo S7 Siemens, que es una implementación estándar del protocolo TCP/IP. Es una librería desarrollada para Python, multiplataforma que permite una comunicación ethernet con PLCs S7 de Siemens. Está basada en la norma ISO TCP (RFC1006). Su diseño está orientado a bloques, donde cada bloque es llamado unidad de datos de protocolo (PDU, por sus siglas en ingles).

Cada transmisión de datos contiene un comando o una respuesta de la misma, este comando tiene los siguientes componentes:

- Una cabecera.
- Un conjunto de parámetros.
- Un dato de parámetros.
- Un bloque de datos.

Los dos primeros elementos están siempre presentes, los otros son opcionales. La biblioteca Snap7 está diseñada de manera que soporte las grandes transferencias de datos industriales a tiempo real. Para cumplir con esto, Snap7Client expone tres características interesantes: la independencia, la transferencia de datos SmartConnect y asíncrona PDU.

La Figura 3.13 muestra los tres objetos que contiene la biblioteca Snap7, éstos son:

1. **Client:** Su función principal es leer/escribir todas las variables del PLC (entrada, salida, db, marcas, temporizadores, contadores).
2. **Server:** Su función es aceptar enlaces S7 por los clientes externos, y las respuestas a sus peticiones.

3. **Partner:** Permite crear una comunicación S7 punto a punto. Cada una de las partes puede enviar los datos de forma asíncrona.

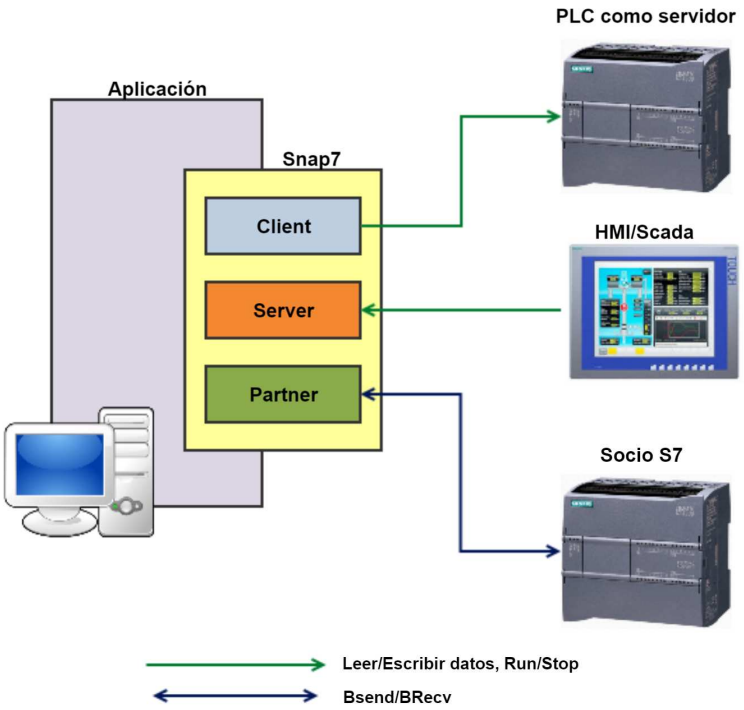


Figura 3.13: Objetos contenidos en la librería Snap7

Para el desarrollo del software se utilizó el objeto *Client* de la librería Snap7, el cual permite la lectura y escritura de datos del PLC. Utilizando este objeto, el dispositivo de campo funciona como servidor.

La Figura 3.14 muestra la arquitectura de un PLC que actúa como servidor. Los cuales se manejan de forma automática por el firmware del procesador de comunicaciones (CP, por sus siglas en inglés), por esta razón no es necesario ninguna configuración en el PLC.

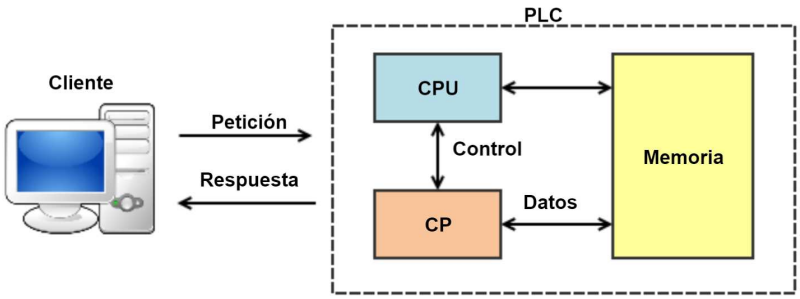


Figura 3.14: Manejo automático de los recursos del servidor

Las funciones más importantes de la librería son:

A. Establecer conexión

Para la comunicación con el PLC, se necesita instanciar un cliente para iniciar el servidor, esto se realiza mediante la función snap7.client.Client().

Los parámetros necesarios para establecer la conexión, además de la dirección IP, son el *rack* (0..7) y el *slot* (1..31). La Tabla 3.4 indica los valores de *rack* y *slot* que se utilizan para los modelos compatibles con esta librería. Para el PLC modelo S7-1200 es posible usar los valores de *rack*=0 y *slot*=0 o *rack*=0, *slot*=1 [27].

Tabla 3.4: Modelos que soporta la librería Snap7

Modelo	**rack**	**slot**	**Observaciones**
S7-300 CPU	0	2	Siempre
S7-400 CPU	No fijo		Siga la configuración del hardware
S7-1200 CPU	0	0	ó 0, 1
S7-1500 CPU	0	0	ó 0, 1

B. Lectura de las variables del PLC

La función que permite la lectura de la memoria del PLC es:

$$plc.read\_area(area, start, length) \tag{3.1}$$

donde:

- **area=** Variable para ubicar en memoria las *tags* del PLC.
- **start=** Indica donde se empezará a leer la variable seleccionada.
- **length=**Longitud en bits de la lectura.

Los valores del área para cada tipo de dato se muestran en la Tabla 3.5.

Tabla 3.5: Ubicación en memoria de los tipos de datos del PLC

Valor	Significado	Nomenclatura
0x81	Proceso de Entrada	I
0x82	Proceso de Salida	Q
0x83	Indicador	M
0x84	DB	No definida
0x1C	Contadores	No definida
0x1D	Temporizadores	No definida

La Tabla 3.6 indica el valor que tiene el parámetro length, dependiendo del tipo de dato que se requiera leer.

Tabla 3.6: Longitud en bits de la lectura

Valor	Tamaño
bit	1
byte	1
word	2
double word	4
real	4
contador	2
temporizador	2

Las representaciones de los tipos de datos en un PLC están estandarizados para que el usuario reconozca fácilmente la variable que desea manipular. Por esta razón, para indicar el tamaño de la variable se utiliza la nomenclatura de la Tabla 3.7, y para el tipo de dato se utiliza la Tabla 3.5.

La Figura 3.15 indica cómo se tiene que definir las *tags* del PLC en la función implementada para la lectura de datos. En este ejemplo se procede a la lectura de la variable de entrada del bit I0.4.

Tabla 3.7: Nomenclatura para el tamaño de las variables del PLC

Significado	Nomenclatura
bit	x
byte (8 bits)	b
word (16 bits)	w
double word (32 bits)	d
real (32 bits float)	freal
contador (16 bits)	No definida
temporizador (16 bits)	No definida

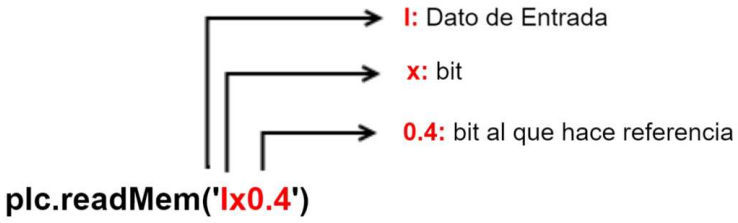


Figura 3.15: Lectura de la entrada del bit I0.4

C. Escritura de las variables del PLC

La función que permite la escritura de la memoria del PLC es:

$$plc.write\_area(area, start, data) \tag{3.2}$$

donde:

- **area=** Variable para la ubicación del área en memoria de las *tags* del PLC.
- **start=** Ubicación en donde se empezará a leer la variable seleccionada.
- **data=** Dirección del *buffer user*(Puntero al área de memoria).

La Figura 3.16 muestra un ejemplo detallado de la función implementada para la escritura del dato proveniente del proceso de salida del bit Q0.0.

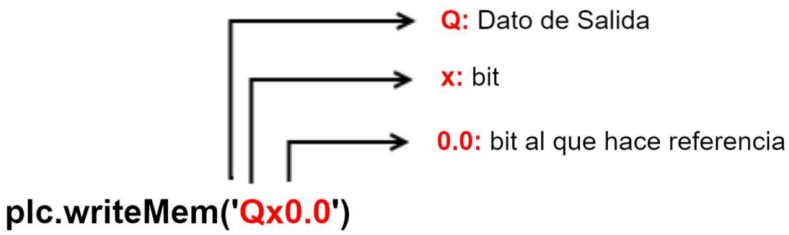


Figura 3.16: Escritura en la variable de salida correspondiente al bit Q0.0

Comunicación con los clientes

La comunicación cliente-servidor se lo realizó mediante *sockets*. Los *sockets* son puntos finales donde uno o varios procesos localizados en diferentes máquinas de una subred intercambian información [28].

Para que la conexión quede establecida existen 3 recursos necesarios, estos son:

- Un protocolo para la comunicación, los más utilizados son: el protocolo de control de transmisión (TCP, por sus siglas en inglés) y el protocolo de datagrama de usuario (UDP, por sus siglas en inglés).
- Un identificador de red. Si se utiliza el protocolo TCP/IP se requiere la dirección IP de la computadora.
- El número de puerto por el cual se enviará la información.

El funcionamiento de esta comunicación es:

- **Servidor:** Se ejecuta sobre una computadora dentro de una subred. Su ejecución inicia al levantar un servicio, mediante un *socket*, en un puerto específico. Cuando el servicio está disponible, escucha a los usuarios que desean acceder a sus recursos mediante peticiones [29].
- **Cliente:** Accede al servidor mediante su dirección IP y el puerto utilizado. Pueden existir múltiples clientes accediendo a él o los servicios brindados por el servidor.

La Figura 3.17 muestra el modelo de comunicación de *sockets* utilizado en Python. Los métodos *send* y *recv* permiten el intercambio de información con los clientes (SCADA y MATLAB).

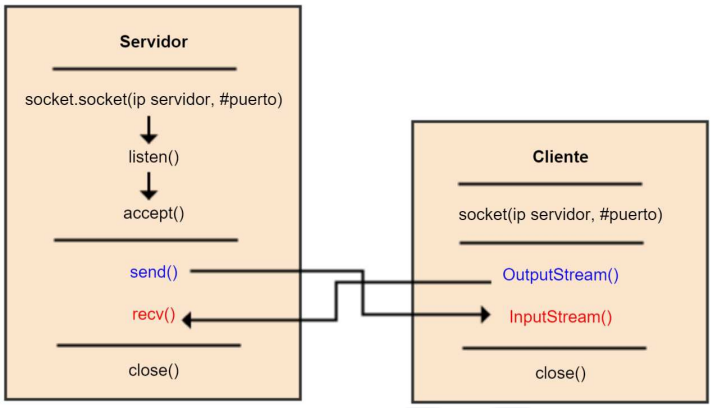


Figura 3.17: Modelo de comunicación entre el servidor Python y el cliente Java

Implementación del servidor

Los requisitos previos para implementar y ejecutar el servidor se muestra en el Apendice A. El servidor se desarrolló en Python 3.5.0, este lenguaje de programación posee una licencia de código abierto. El software está diseñado por clases, las cuales se describen a continuación.

A. Clase Main

La Figura 3.18 muestra los métodos y atributos que dispone la clase `Main`. En esta clase se genera la interfaz del servidor y envía los parámetros que se requieren para el inicio del programa. Los atributos que posee son:

- **ventana:** Se utiliza para instanciar la ventana principal del programa. Esta variable permite agregar los diferentes botones y cajas de texto, presentes en la interfaz principal.
- **aux_obs:** Es una variable auxiliar que indica el estado actual del servidor. En caso de fallo, o desconexión del cliente, se mostrará un mensaje en la ventana principal.

- **aux_salida_hilo_obs:** Los procesos de lectura y escritura de las variables del PLC deben trabajar de manera independiente, por esta razón, se implementan hilos en el software. Estos hilos son controlados con esta bandera, indicando su estado. Si el cliente se desconecta del sistema, esta variable indica el error que se generó mediante la variable aux_obs.

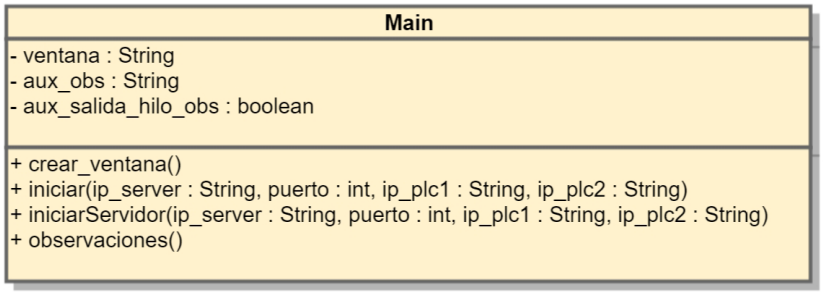


Figura 3.18: Métodos y atributos de la clase `Main`

La clase `Main` posee diferentes métodos. La Figura 3.18 muestra los 6 métodos más importantes, éstos son:

- **crear_ventana:** La ventana principal y los valores iniciales de cada caja de texto son cargados en este método.
- **iniciar:** Al presionar el botón « conectar », que se encuentra en la ventana principal, este método es invocado. Sus objetivos principales son: ejecutar los hilos que inician el servidor, el reporte de observaciones y validar que los parámetros ingresados sean los correctos.
- **iniciar_servidor:** Este método se ejecuta dentro de un hilo. Si los parámetros ingresados en la ventana principal son correctos, este método se ejecuta dentro del programa.
- **observaciones:** Las excepciones generadas por diferentes fallos dentro del programa son capturadas y mostradas mediante este método.

B. Clase Servidor

La Figura 3.19 muestra los métodos y atributos que dispone la clase `Servidor`. Esta clase inicia la lectura de las variables del PLC, y paralelamente guarda en la base de datos MySQL los eventos sucedidos en la planta. Las funciones principales del servidor están contenidas en esta clase, éstas son:

- El control de los datos enviados a los usuarios.
- Contiene los parámetros de inicio de comunicación con los clientes SCADA y MATLAB.
- Detección de errores sucedidos en la comunicación con los clientes.

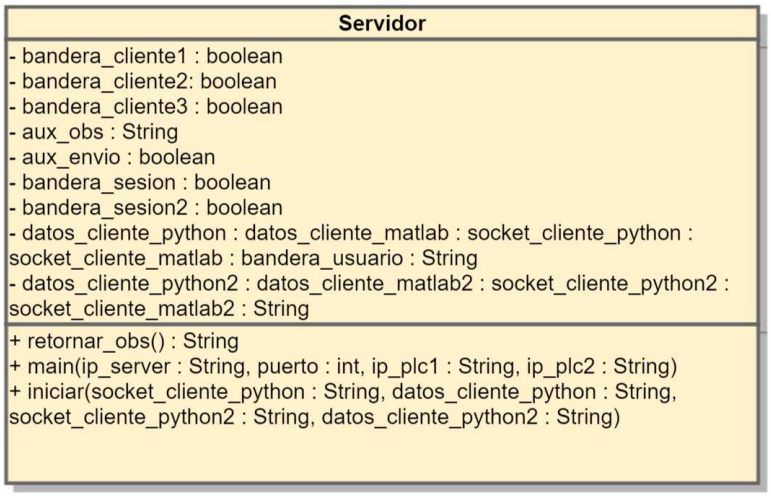


Figura 3.19: Métodos y atributos de la clase `Servidor`

Los atributos que posee son:

- **bandera_cliente1, bandera_cliente2, bandera_cliente3:** Estas banderas indican el cierre de sesión de los 3 usuarios disponibles en el sistema.
- **aux_obs:** Permite conocer el estado de la comunicación entre el cliente y el servidor. Si existe algún problema, éste será mostrado en la interfaz del servidor.

- **aux_envio:** Indica el envío de datos a los clientes, ya que estos se han autenticado correctamente.
- **bandera_sesion, bandera_sesion2:** Estas banderas se activan cuando el *socket* establece una conexión con un cliente. El socket espera hasta establecer una comunicación con un cliente SCADA y con un cliente MATLAB. Los parámetros de conexión son guardados en las variables datos_cliente y socket_cliente.

La Figura 3.19 muestra los 3 métodos más importantes de la clase `Servidor`, éstos son:

- **retornar_obs():** Indica el estado actual de la conexión entre el servidor y los clientes.
- **main():** Sus funciones principales son:
  - Crear el socket para la comunicación.
  - Iniciar los hilos para leer y guardar los datos de cada PLC.
  - Espera la conexión de un cliente al servidor.
- **iniciar():** Inicia el envío de los datos, dependiendo del usuario que se autentico.

C. Clase Sesión

La Figura 3.20 muestra los métodos y atributos que dispone la clase `Sesión`. Los atributos de esta clase permiten conocer cuando un cliente inicia una comunicación con el servidor.

Los métodos que tiene la clase `Sesión` son:

- **crear():** Inicia la comunicación cliente-servidor y espera hasta que otros clientes accedan a este servicio.

- **getclientes():** Retorna los parámetros de cada cliente para el envío de los datos.

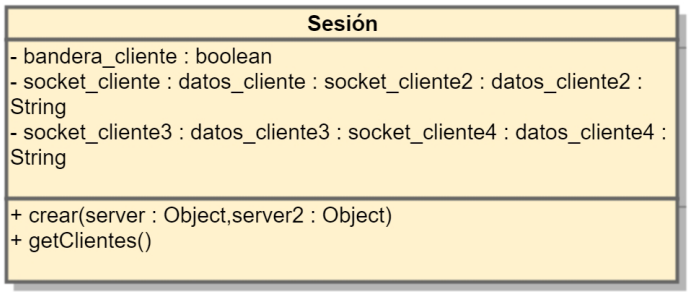


Figura 3.20: Métodos y atributos de la clase `Sesión`

D. Clase Usuario

La Figura 3.21 muestra los métodos y atributos que dispone la clase `Usuario`. El objetivo de esta clase es la autenticación de cada usuario que accede al PLC.

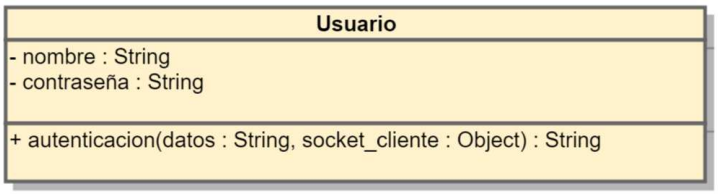


Figura 3.21: Métodos y atributos de la clase `Usuario`

E. Clase Encriptación

La Figura 3.22 muestra los métodos y atributos de la clase `Encriptación`. La seguridad en el envío de datos es muy importante ante posibles ataques. Por esta razón, se implementó un canal seguro utilizando un cifrado DES. El método codificar realiza esta encriptación.

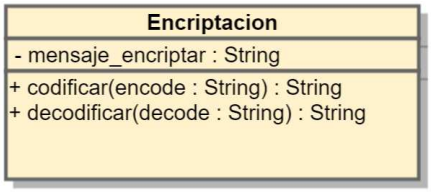


Figura 3.22: Métodos y atributos de la clase `Encriptación`

F. Clase Conexión

La Figura 3.23 muestra los métodos que dispone la clase `Conexión`. Es la parte principal del servidor, ya que interactúa con la librería Snap7 de Python. Los métodos que tiene esta clase son:

- **readMem():** Realiza la lectura de los estados de las variables del PLC, el valor dependerá de las tags recibidas como parámetro en el método.
- **writeMem():** Cambia el estado de las variables de salida del PLC.

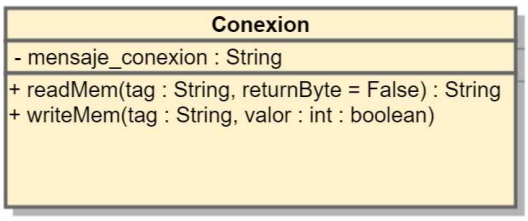


Figura 3.23: Métodos y atributos de la clase `Conexión`

G. Clase Base de Datos

La Figura 3.24 muestra los métodos de los que dispone la clase `Base de Datos`. Cada método ejecuta una conexión con el gestor de base de datos MySQL *Community Server*. Se utiliza este servidor por tener licencia pública general (GPL por sus siglas en inglés). La función que realiza cada método es:

- **insertar_PLC1():** Guarda el cambio de estado que tiene cada variable del PLC. Accede a la base de datos del PLC1.

- **insertar_PLC2():** Tiene la misma función del método anterior, y accede a la base de datos del PLC2.
- **mostrar_PLC1():** Retorna una matriz con los datos que se generan de las consultas que se realicen en el PLC1.
- **mostrar_PLC2():** Retorna una matriz con los datos que se generan de las consultas que se realicen en el PLC2.

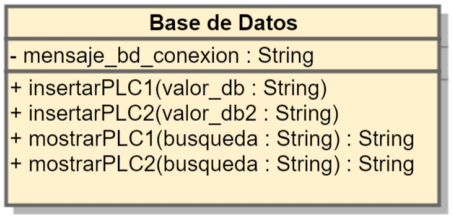


Figura 3.24: Métodos y atributos de la clase `Base de Datos`

Cada base de datos es creada con anterioridad en MySQL Workbench 6.3 CE.

H. Clase Manejo de Variables

La Figura 3.25 muestra los atributos y métodos que tiene la clase `Manejo de Variables` del PLC1. Este método se ejecuta para los 3 usuarios que se tienen en el sistema SCADA. Sus funciones principales son:

- Reportar el estado de la conexión con los usuarios que se ejecutan en ese instante.
- Guardar los eventos que suceden en cada usuario conectado.
- Enviar los datos actualizados de las variables del PLC.
- Recibir órdenes de los clientes y ejecutar sus peticiones. Éstas son: alarmas revisadas, generación de reportes, envío de datos históricos.

Los atributos que posee son:

- **plc:** Objeto que contiene los parámetros de conexión del PLC. Este objeto permite leer y escribir los datos del PLC.
- **bandera:** Indica que los datos han sido leídos e inicie el proceso de guardado en la base de datos.
- **bandera1:** Bandera que permite finalizar el hilo del método de escritura.
- **bandera2:** Bandera que finaliza el hilo que permite guardar en la base de datos.
- **bandera3:** Bandera que termina el proceso de recepción de datos con el cliente.
- **aux_error:** Indica que existió un error en la comunicación con el cliente.
- **aux_db..aux_bd11:** Variables globales que contienen los estados de todos los proceso de entrada y salida del PLC.
- **bandera_envio:** Da a conocer a la clase que existe una conexión con un cliente y permite enviar los datos al usuario conectado.
- **aux_envio:** Esta bandera se activa solo para guardar el primer estado de las variables del PLC, luego verifica el cambio de estado de los procesos de entrada y salida.
- **a1_db..a13_db:** Contiene el estado anterior de cada variable del PLC.

La clase `Manejo de Variables` tiene diferentes métodos, éstos son:

- **getsesion():** Reporta el cierre de sesión del usuario y el estado de conexión del mismo.
- **getenvio():** Este método indica el envío de datos al cliente, obteniendo los parámetros de conexión del usuario.
- **getplc1():** Retorna todos los estados de las variables del PLC actualizadas.

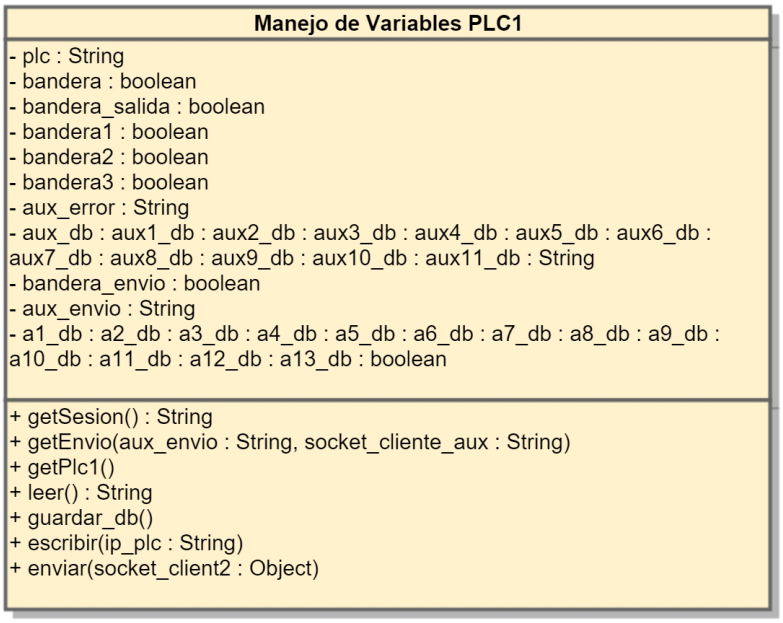


Figura 3.25: Métodos y atributos de la clase `Manejo de Variables PLC1`

- **leer():** Obtiene los datos del PLC y retorna estos valores en un *String* concatenado.
- **guardar_db():** Almacena en la base de datos los valores actuales del dispositivo de campo.
- **escribir():** Inicia los procesos de lectura y escritura del PLC cada 3 segundo y verifica el estado de la variable bandera_envio.
- **enviar():** Recibe las solicitudes enviadas desde el cliente, dando respuesta a las mismas.

I. Clase Conexión MATLAB

La Figura 3.26 muestra los atributos y métodos que tiene la clase `Conexión MATLAB`. Su función principal es establecer una conexión con el cliente MATLAB. Los atributos que tiene esta clase son:

- **bandera:** Inicia la ejecución del hilo para el envío de datos a MATLAB.

- **bandera_cliente:** Indica el cierre de sesión del cliente MATLAB.

En la Figura 3.26 indica los métodos de la clase `Conexión MATLAB`, éstos son:

- **autenticar():** Espera que el usuario ingrese sus datos de acceso correctamente, para que se envíen los datos.
- **enviar():** Responde a las peticiones que el cliente MATLAB realiza al servidor.

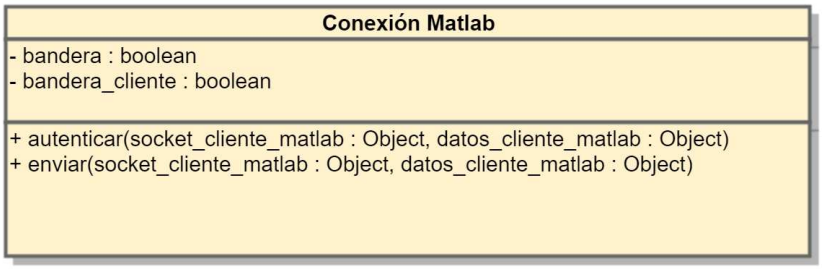


Figura 3.26: Métodos y atributos de la clase `Conexión MATLAB`

En el Apendice B.1 se muestra la asociación que existe entre las clases implementadas en el servidor.

3.4. Desarrollo de la aplicación cliente

El cliente se desarrolló en Java, este es un lenguaje de programación de código abierto. Antes de comenzar con la implementación se analizó la jerarquía de las principales interfaces y clases para evitar problemas futuros, y así desarrollar una lógica de funcionamiento de nuestra aplicación cliente.

La Figura 3.27 muestra la jerarquía de las principales interfaces de la aplicación cliente. A continuación, se detallará como se implementó cada una de las interfaces.

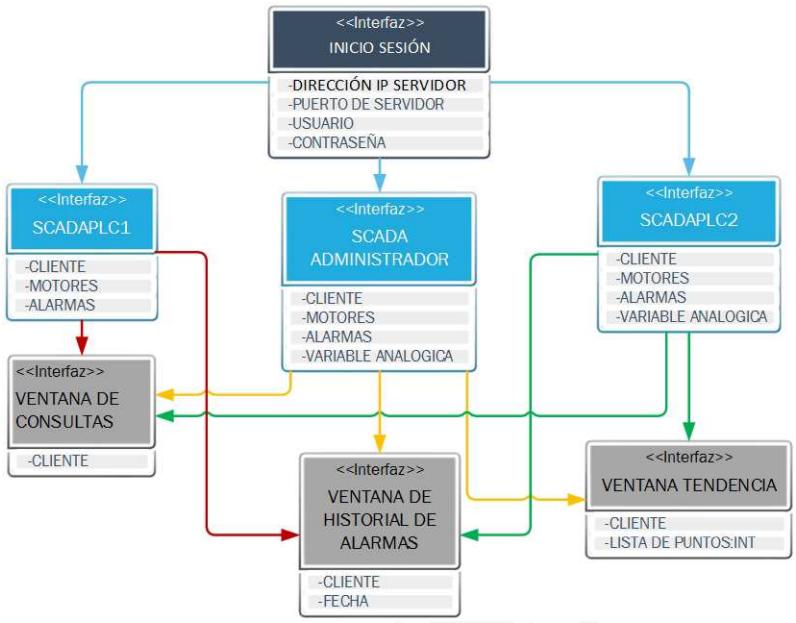


Figura 3.27: Jerarquía de las principales interfaces de la aplicación cliente

3.4.1. Clase `V_Inicio`

La clase `V_Inicio` hace referencia a la ventana de inicio ó inicio de sesión. La ventana de inicio es la clase padre de nuestra aplicación cliente. En esta ventana se puede realizar la conexión con el servidor. Además, se necesita el usuario y contraseña para realizar la autenticación.

La Figura 3.28 muestra la ventana de inicio, donde se muestran tres botones:

1. **Botón « conectar »:** Su función es iniciar la conexión con el servidor, abriendo un puerto de la aplicación cliente mediante la utilización de un socket.

2. **Botón « aceptar »:** Su función es obtener el usuario y contraseña de la ventana de inicio. Después se envía al servidor, que comprueba si los datos son correctos. El servidor responderá con un mensaje si los datos son correctos o incorrectos.

3. **Botón « cerrar »:** Su función es cerrar la aplicación del cliente.

Figura 3.28: Ventana de inicio

La Figura 3.29 muestra los parámetros y métodos de la clase v_Inicio.

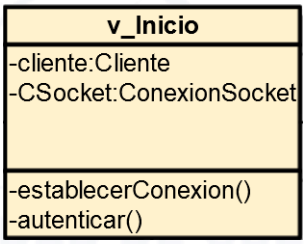


Figura 3.29: Clase `V_Inicio`

Los parámetros de la clase `v_Inicio` son los siguientes:

- **cliente:** Es la variable donde se guarda el usuario y contraseña del usuario identificado en la aplicación.
- **CSocket:** Es la variable tipo *ConexionSocket* usada para gestionar la conexión.

Los métodos de la clase `v_Inicio` son los siguientes:

- **establecerConexion:** Define la conexión con el servidor.
- **autenticar:** Realiza la identificación del usuario en la aplicación.

Las variables tipo cliente se conforman de un usuario y una contraseña. En la ejecución de la aplicación cliente se maneja una variable de este tipo. El cliente que realiza la autenticación con el servidor de forma correcta se guarda en toda la ejecución de la aplicación, esta variable se obtiene de la ventana de inicio.

Este cliente es heredado por las demás interfaces. La herencia de atributos se muestra en la Figura 3.27, esto se denota con flechas desde la clase padre a la clase hijo. En el caso de las clases que heredan de la interfaz inicio de sesión se muestra con flechas azules.

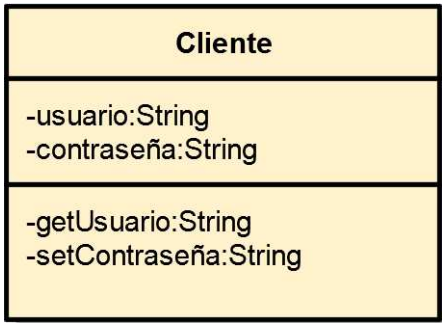


Figura 3.30: Clase `Cliente`

La Figura 3.30 muestra como está conformada la clase `Cliente` con sus atributos y métodos.

La aplicación cliente está orientada al uso de cuentas (usuario y contraseña). Las tres interfaces del nivel inferior, mostradas en la Figura 3.27 con color azul, se presentan de acuerdo a los clientes autenticados.

Las variables tipo `ConexionSocket` se conforma de un *host* y un puerto TCP. En la ejecución de la aplicación se utiliza un objeto de este tipo para gestionar las conexiones. Este objeto es heredado por las demás interfaces. La Figura 3.31 muestra la clase `ConexionSocket` con sus respectivos métodos y parámetros.

La clase `ConexionSocket` tiene los siguientes parámetros:

- **host:** Es la dirección IP del servidor a conectarse.

- **nPuerto:** Es el número de puerto de la aplicación servidor.

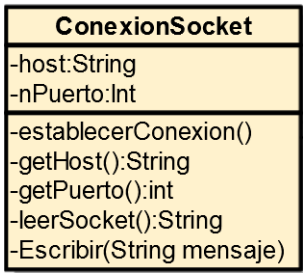


Figura 3.31: Clase `ConexionSocket`

La clase `ConexionSocket` tiene los siguientes parámetros:

- **EstablecerConexion:** Define la conexión con el servidor mediante el uso de la dirección IP y el puerto.
- **getHost:** Devuelve como resultado el host de la conexión.
- **getPuerto:** Devuelve como resultado el puerto de la aplicación servidor.
- **leerSocket:** Realiza la lectura del *buffer* del socket de conexión.
- **Escribir:** Realiza la escritura de un mensaje en el socket de conexión.

Las interfaces SCADA implementadas en la aplicación cliente están asociadas a la simulación de procesos sencillos como la monitorización de los estados de los motores de la planta, activación de alarmas y nivel de llenado del silo de almacenamiento. Para ello se implementaron dos: la clase `Motor` y la clase `Alarma`.

La Figura 3.32 muestra la clase `Alarma` con sus atributos y métodos. Los atributos de esta clase son las características principales de una alarma, donde se puede ejecutar los métodos de acuerdo a los permisos del cliente identificado en la aplicación.

Los atributos de esta clase son los siguientes:

- **estado:** Hace referencia a si la alarma está encendida o apagada.

- **nombre** e **id**: Permiten identificar con exactitud una alarma.
- **grupo:** Permite identificar si la alarma es del PLC1 o PLC2.
- **origen:** Es un mensaje para informar las circunstancias de activación de la alarma.
- **tipo:** Alerta al operador sobre las situaciones anómalas presentes en el sistema e implican una intervención [30].
- **momento:** Es una variable tipo fecha que se obtiene del sistema operativo del computador cuando la alarma se activa.

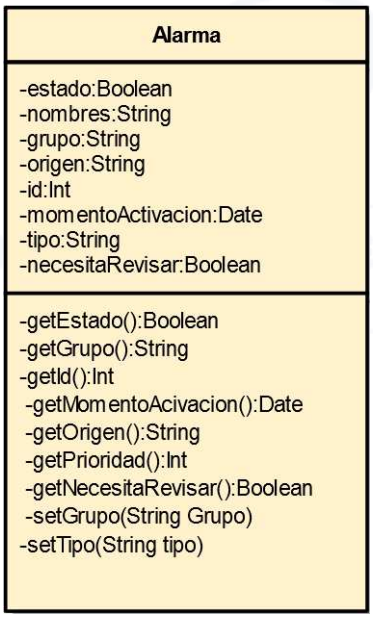


Figura 3.32: Clase `Alarma`

Los métodos de esta clase son los siguientes:

- **getEstado:** Devuelve el estado de la alarma.
- **getGrupo:** Devuelve a que grupo pertenece la alarma.
- **getOrigen:** Devuelve el origen de la alarma.

- **getId:** Devuelve el id de la alarma.
- **getMomentoActicacion:** Devuelve la fecha y hora de activación de la alarma.
- **getNecesitaRevisar:** Devuelve el estado de revisión de la alarma.

La Figura 3.33 muestra la clase `Motor` con sus parámetros y métodos. Los parámetros y métodos asociados a esta clase son los que caracterizan a un motor, de acuerdo a su funcionamiento en nuestro sistema SCADA.

Figura 3.33: Clase `Motor`

Los parámetros de esta clase son los siguientes:

- **nombre e id:** Permitirán la identificación del motor.
- **estado:** Describe si el motor esta encendido ó apagado.

Los métodos de esta clase son:

- **getId:** Devuelve el id del motor.
- **getNombre:** Devuelve el nombre del motor.
- **getEstado:** Devuelve el estado del motor.
- **setEstado:** Permite modificar el estado del motor.

3.4.2. Clase `SCADAPLC1`

La clase `SCADAPLC1` esta desarrolla para los procesos que se simulan en el PLC1. La clase tiene dos usuarios U1PLC1 y U2PLC1. El usuario U1PLC1 puede realizar las siguientes acciones:

- Acceso para visualizar los estados de las variables.
- Acción sobre el encendido y apagado de los motores de esta interfaz.
- Realizar el reconocimiento de alarmas en el momento que se genera una alarma, tomando responsabilidad sobre la misma.

El usuario U2PLC1 puede realizar las siguientes acciones:

- Acceso a visualizar los estados de las variables.
- Permiso restringido para modificar el encendido o apagado de los motores.
- Realizar acciones como revisar un historial de las alarmas, generar reportes.
- Realizar el reconocimiento de alarmas en el momento que se genera una alarma, tomando responsabilidad sobre la misma.

La Figura 3.34 muestra la interfaz gráfica donde podemos apreciar tres bloques encerrados en rectángulos:

1. **Bloque de alarmas (rectángulo rojo):** Muestra las alarmas activadas. Permite al usuario hacer la revisión, también muestra un contador de cuantas alarmas están activas.
2. **Bloque de esquema (rectángulo azul):** Muestra el esquema del proceso simulando en el PLC1. La descripción de los componentes involucrados se detalla en el Apéndice F.

3. **Barra de menú de navegación (rectángulo amarillo):** Esta barra consta de: el logo de la institución, menús de tareas implementado mediante botones, hora y fecha.

Los tres bloques anteriormente mencionados están presentes en las clases SCADAPLC1, SCADAPLC2, SCADA ADMINISTRADOR.

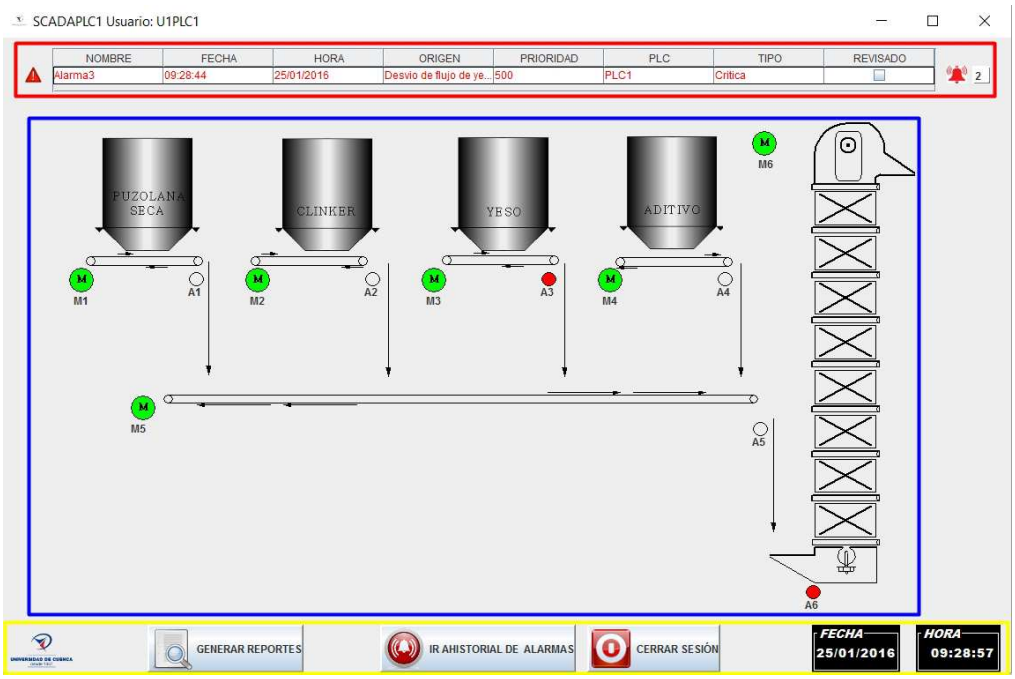


Figura 3.34: Interfaz gráfica para procesos simulados en PLC1

Bloque de alarmas

La Figura 3.35 muestra bloque de alarmas y sus elementos.

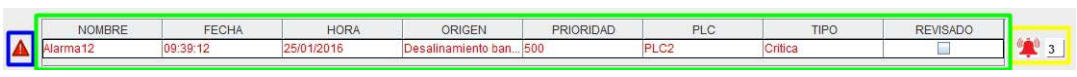


Figura 3.35: Bloque de alarmas

El bloque de alarmas consta de tres elementos, éstos son:

1. Una tabla de una fila con ocho columnas, en donde se cargan los datos de una alarma activa(rectángulo verde, Figura 3.35). En la columna 8 (Revisado) el usuario puede hacer clic para que la aplicación envié los datos que corresponde a la revisión de la alarma.

2. Icono de tipo de alarma (rectángulo azul, Figura 3.35): Este icono hace referencia al tipo de alarma. En la aplicación existe dos tipos de alarma: crítica y de advertencia.

3. El contador de las alarmas (rectángulo amarillo, Figura 3.35): Hace referencias a cuantas alarmas están activas.

La Figura 3.36 muestra los dos tipos de alarmas, con los íconos utilizados en la aplicación cliente.

Figura 3.36: Tipos de alarmas

Las alarmas se clasifican de acuerdo a su origen y como afecta al funcionamiento del sistema.

Bloque de esquema

En este bloque se realizó mediante la asociación de imágenes a etiquetas y botones en la programación de una interfaz gráfica de Java. La interfaz gráfica del SCADA es el conjunto de etiquetas y botones ubicados correctamente sobre un panel. Las imágenes usadas en la interfaz son generadas por programas del tipo CAD. A continuación se listan algunas de las alternativas de código libre que pueden ser utilizadas para este objetivo:

- BRL-CAD
- LibreCAD

- FreeCAD
- gCAD3D
- QCad

Barra de menú de navegación

La Figura 3.37 muestra la barra de menú de la interfaz SCADAPLC1 con sus componentes, éstos son:

- Logo identificativo de la institución (rectángulo rojo).
- Botón « generar reportes » (rectángulo amarillo): Al presionar este botón ejecuta la ventana para generar reportes.
- Botón « ir a historial de alarmas » (rectángulo café): Al presionar este botón ejecuta la ventana de historial de alarmas.
- Botón « cerrar sesión » (rectángulo morado): Al presionar este botón se cierra la sesión y ejecuta la ventana de inicio de sesión.
- Hora y fecha (rectángulo naranja): Muestra la hora y fecha actual obtenida del sistema de la computadora donde se ejecuta la aplicación.

Figura 3.37: Barra de menú de navegación de la interfaz SCADAPLC1

La Figura 3.38 muestra los parámetros y métodos de la clase `SCADAPLC1`.
Los parámetros de esta interfaz son los siguientes:

- **Cliente:** Es el cliente identificado en la aplicación, es heredado de la interfaz inicio de sesión.
- **Alarmas:** Es la lista de alarmas a monitorear en esta interfaz.
- **Motores:** Es la lista de motores a monitorear y controlar en esta interfaz.

- **Bandera Activación:** Hace referencia a la activación de esta interfaz. Su valor cuando la interfaz está ejecutándose es 1 ó verdadero, mientras que si se está ejecutando otra interfaz es 0 o falso.

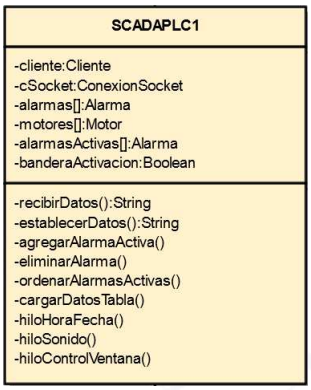


Figura 3.38: Clase `SCADAPLC1`

Los métodos de esta interfaz son los siguientes:

- **RecibirDatos:** Es la ejecución de un hilo que realiza la lectura del *buffer* de entrada del socket TCP/IP y se ejecuta cada 3 segundos.
- **EstablecerDatos:** Este método configura la interfaz de acuerdo a los datos leídos en el método anterior.
- **AgregarAlarmaActiva:** Este método se ejecuta si una alarma se activa.
- **EliminarAlarma:** Este método se ejecuta cuando una alarma ha sido revisada por el usuario.
- **OrdenarAlarmasActivas:** Este método se ejecuta antes de cargar los datos a la tabla.
- **CargarDatosTabla:** Este método se ejecuta cada que se activa una nueva alarma o cuando se hace la revisión de una alarma por parte del operador.
- **HiloHoraFecha:** Se ejecuta cuando la venta se abre. Este método implementa un hilo que se encarga de obtener la fecha y la hora del sistema del ordenador donde se está ejecutando la aplicación.

- **HiloSonido:** Este hilo tiene la función de reproducir el sonido cuando una alarma está activa y no ha sido revisada por el operador.
- **HiloControlVentana:** Este hilo es el encargado de revisar los estados de los motores de la interfaz. Cuando un motor de una banda transportadora este encendido realiza los cambios de imágenes de la banda, dando un efecto dinámico a la interfaz.

3.4.3. Clase `SCADAPLC2`

La interfaz SCADAPLC2 esta desarrolla para los procesos que se simulan en el PLC2. La interfaz está asociada a dos usuarios U1PLC2 y U2PLC2. El usuario U1PLC2 puede realizar las siguientes acciones:

- Acceso para visualizar los estados de las variables.
- Acción sobre el encendido y apagado de los motores de esta interfaz.
- Realizar acciones como revisar un historial de las alarmas, generar reportes y monitorizar la tendencia del comportamiento de la variable analógica.
- Realizar el revisado de alarmas en el momento que se genera una alarma, tomando responsabilidad sobre la misma.

El usuario U2PLC2 puede realizar las siguientes acciones:

- Acceso a visualizar los estados de las variables.
- Permiso restringido para modificar el encendido o apagado de los motores.
- Realizar acciones como revisar un historial de las alarmas, generar reportes y monitorizar la tendencia del comportamiento de la variable analógica.
- Realizar el revisado de alarmas en el momento que se genera una alarma, tomando responsabilidad sobre la misma.

La Figura 3.39 muestra la interfaz gráfica del SCADA para los procesos simulados en el PLC2, donde podemos apreciar tres bloques encerrados en rectángulos:

1. Bloque de alarmas (rectángulo rojo).
2. Bloque de esquema (rectángulo azul). Es la descripción de los componentes involucrados se detallan en el Apéndice F.
3. Barra de menú de navegación (rectángulo amarillo).

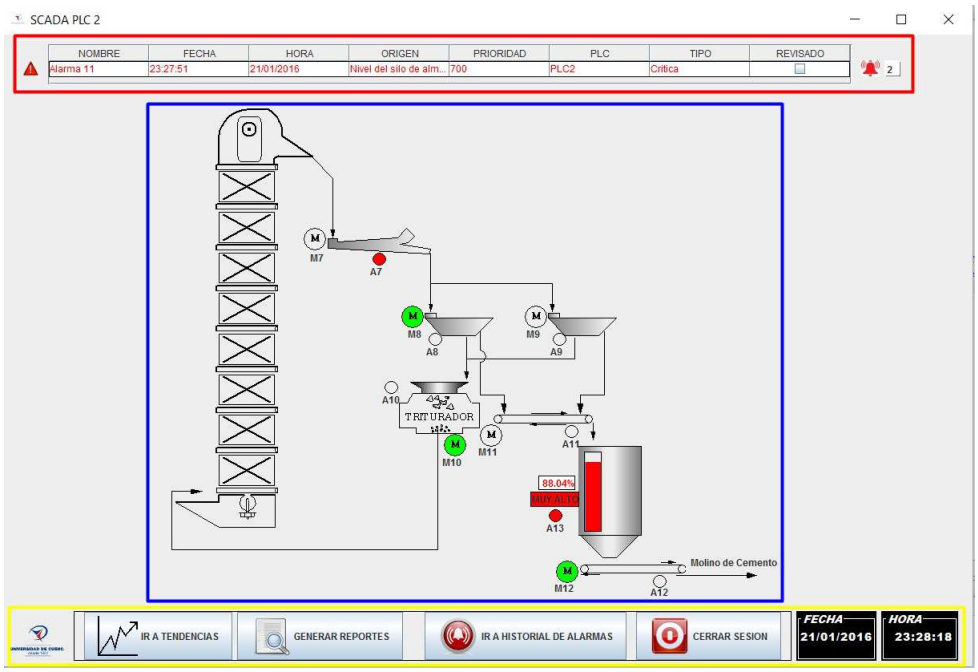


Figura 3.39: Interfaz gráfica para los procesos simulados en el PLC2

La interfaz SCADAPLC2 incluye una tarea adicional a la que se presentó en la interfaz SCADAPLC1; esta tarea es graficar tendencias de una variable analógica. Esta tarea es importante para visualizar el comportamiento de variables analógicas. En este caso, se simula el llenado del silo de almacenamiento a través de un potenciómetro conectado en la entrada analógica del PLC. Esta variación de voltaje puede representar, en otras aplicaciones, señales provenientes de sensores de temperatura, presión, caudal, velocidad, etc.

La Figura 3.40 muestra la barra de menú de la clase `SCADAPLC2` con sus componentes, éstos son:

- Logo identificativo de la institución (rectángulo rojo).
- Botón para visualizar tendencias (rectángulo verde): Al presionar este botón se ejecuta la ventana para visualizar las tendencias de la variable analógica.
- Botón para generar reportes (rectángulo amarillo): Al presionar este botón ejecuta la ventana para generar reportes.
- Botón « ir a historial de alarmas » (rectángulo café): Al presionar este botón ejecuta la ventana de historial de alarmas.
- Botón « cerrar sesión » (rectángulo morado): Al presionar este botón se cierra la sesión y ejecuta la ventana de inicio de sesión.
- Hora y fecha (rectángulo naranja): Muestra la hora y fecha actual obtenida del sistema de la computadora, donde se ejecuta nuestra aplicación.

Figura 3.40: Barra de menú de navegación de la interfaz SCADAPLC2

La Figura 3.41 muestra los parámetros y métodos de la clase `SCADAPLC2`.

El atributo ValoresAnalogicos es un vector donde se guardan los valores de la variable analógica. Los valores de la variable analógica se agregan siempre que se reciban datos desde el servidor.

3.4.4. Clase `SCADA ADMINISTRADOR`

La clase `SCADA ADMINISTRADOR` representa la implementación de una interfaz gráfica donde se pueden apreciar todos los procesos del sistema, proveyendo además al usuario con funcionalidades de control sobre el proceso. También está asociada a dos usuarios que son los siguientes:

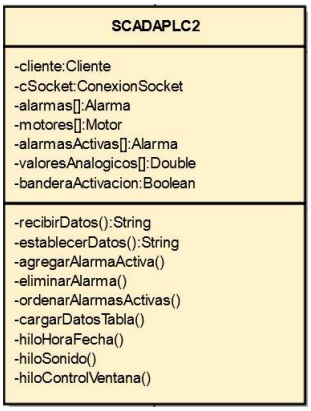


Figura 3.41: Clase `SCADAPLC2`

1. **U3:** Este usuario tiene acceso total sobre las operaciones del sistema SCADA.

2. **U3lock:** Este usuario tiene acceso limitado sobre las operaciones de encendido y apagado de los motores.

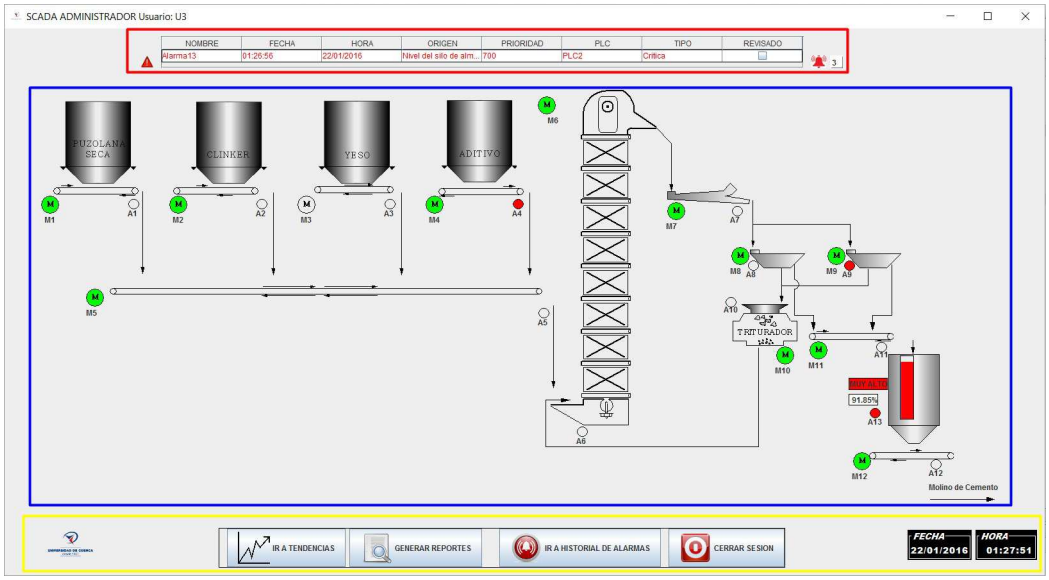


Figura 3.42: Interfaz gráfica de para los procesos simulados en el sistema

La Figura 3.42 muestra la interfaz gráfica del SCADA para los usuarios U3 y U3lock, también podemos apreciar los tres bloques:

1. Bloque de alarmas (rectángulo rojo).
2. Bloque de esquema (rectángulo azul).
3. Barra de menú de navegación (rectángulo amarillo).

La barra de menú de navegación es la misma de la clase `SCADAPLC2`. La Figura 3.43 muestra los parámetros y métodos de la clase `SCADA ADMINISTRADOR`.

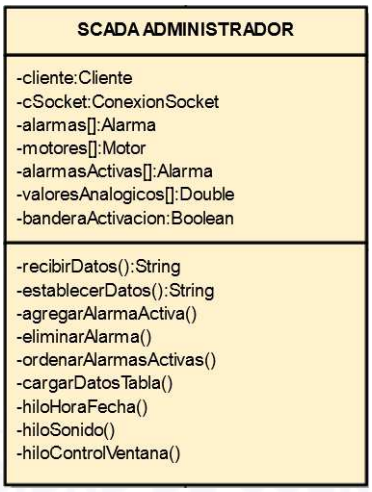


Figura 3.43: Clase `SCADA ADMINISTRADOR`

3.4.5. Clase `V_Reportes`

La clase `V_Reportes` hace referencia a la ventana de reportes, el cual permite generar un reporte. Un reporte se genera a través de una petición al servidor mediante un mensaje TCP/IP. En el mensaje se envía un intervalo de tiempo y el tipo de reporte. El intervalo de tiempo se define por fecha inicial, hora inicial, fecha final y hora final. Para el desarrollo de la interfaz gráfica se utilizó una librería externa, *jcalendar* versión 1.4. Esta librería nos permite utilizar el calendario del sistema operativo donde se ejecuta la aplicación cliente.

La Figura 3.44 muestra la interfaz gráfica de la ventana de reportes. La interfaz gráfica tiene tres bloques para configurar una petición de reporte al servidor, éstos son:

1. El bloque de tipo de reportes, en donde se puede escoger un reporte por fecha, por estados de las variables en el caso de una alarma, por alarmas revisadas y un reporte de toda la base de datos. El procesamiento para realizar un reporte se demora de acuerdo al tipo de reporte que desee el operario.

2. El bloque de datos iniciales, el cual permite configurar una fecha inicial y una hora final.

3. El bloque de datos finales, el cual permite configurar una fecha final y una hora final.

Figura 3.44: Interfaz gráfica de la ventana de reportes

La interfaz gráfica contiene dos botones:

- El botón « generar reporte », al ser presionado hace la petición de generar reporte.

- El botón « salir », al ser presionado cambia el estado de la bandera de activación y cierra la ventana de reportes.

La Figura 3.45 muestra los parámetros y métodos de la ventana de reportes. Los parámetros fecha inicial y fecha final son variables del tipo *date*, se obtiene de la interfaz gráfica al presionar el botón « consultar ». El parámetro banderaActivacion sirve para controlar el hilo de lectura del reporte.

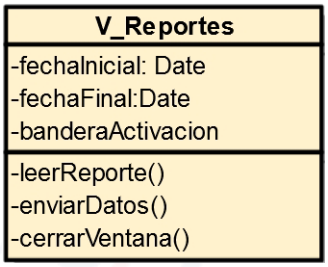


Figura 3.45: Parámetros y métodos de la clase `V_Reportes`

Los métodos de la clase `V_Reportes` son los siguientes:

- **LeerReporte:** Este método genera un hilo que recibe los archivos enviados por el servidor.
- **EnviarDatos:** Este método envía el mensaje de petición de reporte al servidor.
- **CerrarVentana:** Este método se ejecuta al presionar el botón « cerrar », su función es cambiar el estado de la bandera de activación y cerrar la interfaz gráfica.

3.4.6. Clase `V_HistorialAlarmas`

La clase `V_HistorialAlarmas` hace referencia a la ventana de historial de alarmas que permite al usuario visualizar los datos de alarmas en un día específico. El usuario debe realizar una petición de los datos al servidor, luego de recibir éstos, la aplicación los carga en la tabla de la interfaz gráfica.

La Figura 3.46 muestra la interfaz gráfica de la ventana de historial de alarmas. La interfaz gráfica tiene una tabla para visualizar los datos, un botón *combo box* que permite elegir una alarma para mostrar su historial del día, dos botones para realizar una petición al servidor y otro para cerrar la ventana.

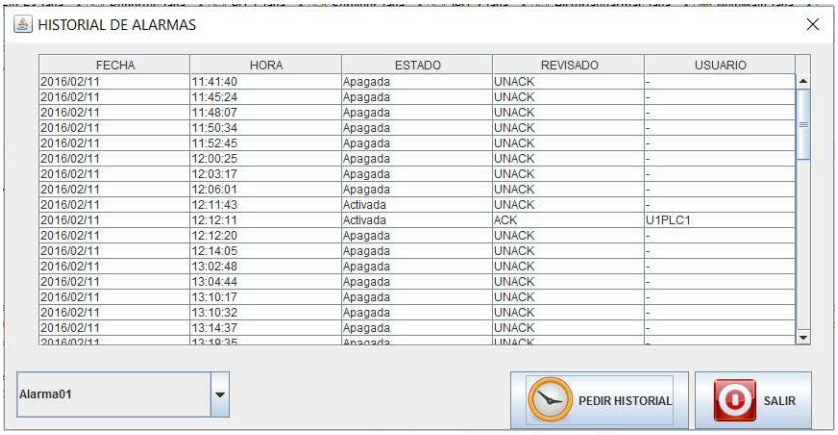


Figura 3.46: Interfaz gráfica de la ventana de historial de alarmas

La Figura 3.47 muestra los parámetros y métodos de la clase `V_HistorialAlarmas`.

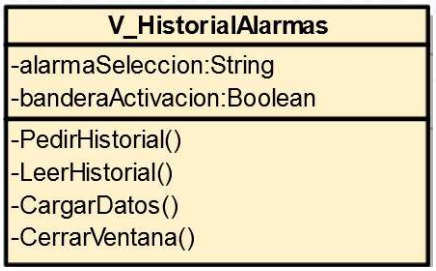


Figura 3.47: Parámetros y métodos de la clase `V_HistorialAlarmas`

Los parámetros de la ventana de historial de alarmas son los siguientes:

- **alarmaSeleccion:** Este parámetro hace referencia a la alarma seleccionada en la interfaz, se obtiene el valor de la interfaz gráfica en el momento de hacer la petición al servidor.
- **banderaActivacion:** Este parámetro toma un valor lógico verdadero en el momento de activación de la interfaz gráfica y cambia a falso en el momento que se cierra la interfaz. Este parámetro permite realizar el control del hilo

de la recepción de datos.

Los métodos de esta clase son los siguientes:

- **PedirHistorial:** Ejecuta la petición del historial de una alarma al servidor, al momento de presionar el botón « pedir historial ».
- **LeerHistorial:** Ejecuta un hilo de lectura del historial a través del socket.
- **CargarDatos:** Ejecuta la tarea de cargar los datos leídos en la tabla de la interfaz gráfica.
- **CerrarVentana:** Cierra la interfaz cambiando el estado de la bandera de activación al valor lógico falso, este método se ejecuta al presionar el botón « cerrar ».

3.4.7. Clase V_Tendencias

La clase `V_Tendencias` hace referencia a la ventana de tendencias, permite al usuario visualizar una gráfica en tiempo real o de un intervalo de tiempo del comportamiento del silo de almacenamiento. La ventana de tendencias utiliza una librería externa de java llamada *jfreechart* utilizada para realizar las gráficas.

Las gráficas de tendencias de variables de proceso, que se monitorizan en un sistema SCADA, permiten tomar decisiones pertinentes a la operación. Las gráficas de tendencias, con registros históricos de eventos de operación permiten además el diseño de sistemas de detección y diagnóstico de fallas.

La interfaz gráfica de esta clase se divide en dos pestañas: la primera para la visualizar una gráfica en tiempo real, y otra para graficar un historial de comportamiento de la variable analógica.

La Figura 3.48 muestra la interfaz gráfica para visualizar el comportamiento de llenado del silo de almacenamiento en tiempo real. Esta interfaz permite pausar la actualización de la variable, de forma que se puedan analizar segmentos específicos de los datos.

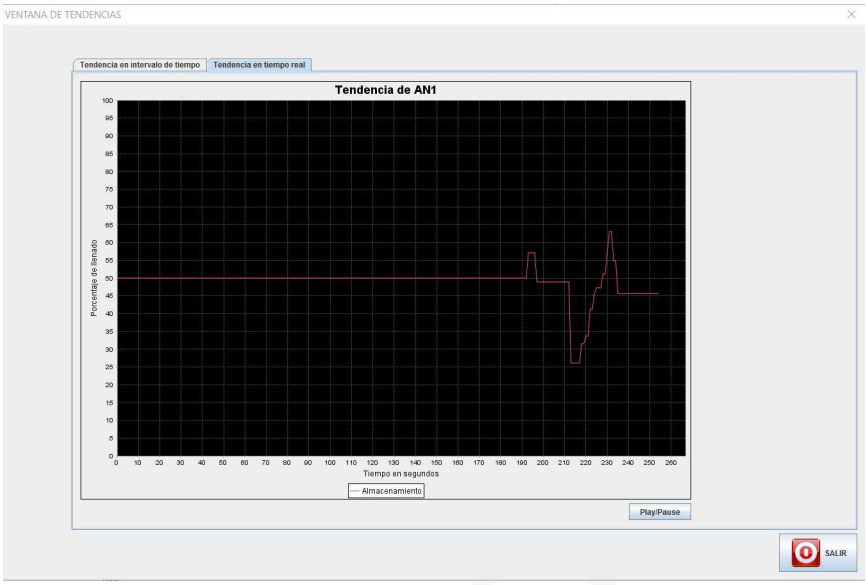


Figura 3.48: Interfaz para graficar en tiempo real

La Figura 3.49 muestra la interfaz gráfica para visualizar la tendencia de un historial de comportamiento del silo de almacenamiento de un intervalo de tiempo determinado por el usuario.

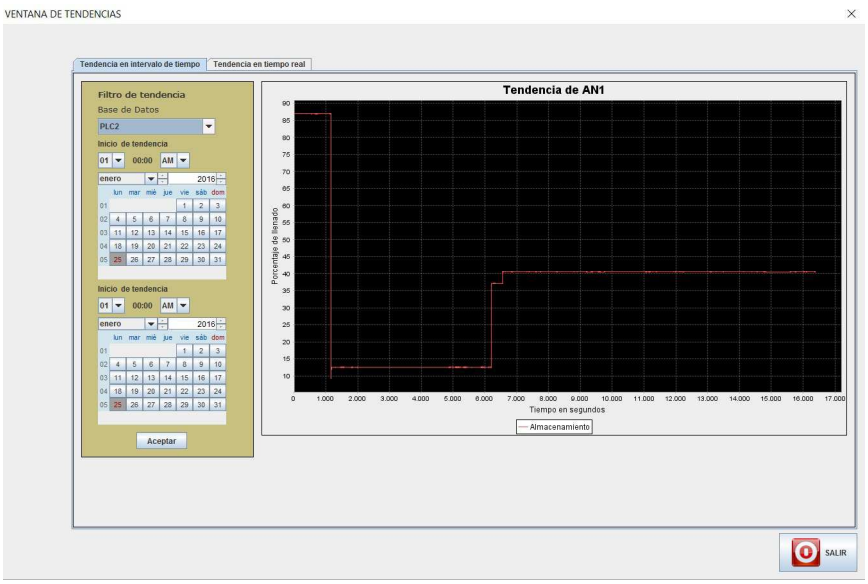


Figura 3.49: Interfaz para graficar un historial

La Figura 3.50 muestra los métodos y parámetros de la clase `ventana` de

tendencias.

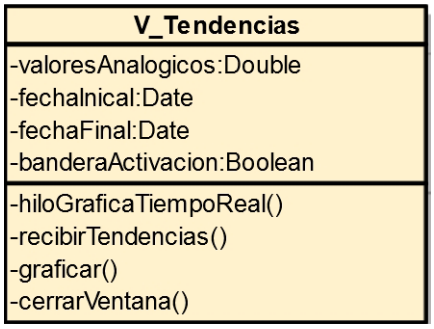


Figura 3.50: Parámetros y métodos de la ventana de tendencias

Los parámetros de la clase ventana de tendencias son:

- **ValoresAnalogicos:** Es una variable global que guarda los datos de la variable analógica en la clase SCADA PLC2 o SCADA ADMINISTRADOR.
- **fechaInicial:** Es la fecha y hora inicial del intervalo de tiempo para hacer la petición del histórico al servidor. Este parámetro se usa en la interfaz para graficar un historial.
- **fechafinal:** Es la fecha y hora final del intervalo de tiempo para hacer la petición del histórico al servidor. Este parámetro se usa en la interfaz para graficar un historial.
- **baderaActivacion:** Permite tener un control de los hilos de ejecución de la interfaz. Esta bandera es verdadera cuando se ejecuta la ventana de tendencias, mientras que cambia a falso cuando se cierra la interfaz gráfica, terminando la ejecución de los hilos.

Los métodos de la ventana de tendencias son:

- **HiloGraficaTiempoReal:** Ejecuta un hilo para graficar los valores analógicos guardados. Este método es generado cada segundo.
- **RecibirTendencias:** Es un hilo que ejecuta la lectura del socket de conexión, recibe los datos del histórico consultado al servidor.

- **Graficar:** Este método se ejecuta en el hilo para graficar a tiempo real y después de recibir los datos del servidor.

En el Apendice B.2 se muestra la asociación que existe entre las principales clases implementadas en el cliente Java.

3.5. Desarrollo del cliente MATLAB

El desarrollo de un cliente en MATLAB permite demostrar las fortalezas del sistema, al comunicarse con software de licenciamiento comercial. El cliente en MATLAB consta de dos interfaces:

- La ventana de inicio en la cual se establece la conexión y se realiza la autenticación con el servidor.
- La interfaz HMI, para el control y monitoreo de las variables de los PLCs.

La Figura 3.51 muestra la ventana de inicio del cliente MATLAB. En la ventana inicio se llenan los datos para establecer conexión y autentificarse con el servidor. Al presionar el botón « conectar » se realiza la apertura del socket correspondiente a la dirección IP y al puerto donde se aloja la aplicación del servidor. Al presionar el botón « ingresar » se realiza la autenticación con el servidor, enviando los datos de usuario y contraseña obtenidos de la interfaz gráfica. Al presionar el boton « salir » se cierra la aplicación cliente de MATLAB.

La ventana principal se cierra al recibir el mensaje de confirmación de usuario autentificado, luego abre la interfaz HMI_MATLAB. El botón « iniciar lectura » permite recibir los datos del sistema. La aplicación cliente MATLAB es un HMI donde es posible visualizar los estados de las variables de los dos PLCs.

La Figura 3.52 muestra la interfaz gráfica del HMI desarrollado en MATLAB, donde se visualiza el estado de las variables y una gráfica de tendencias de la variable analógica del sistema.

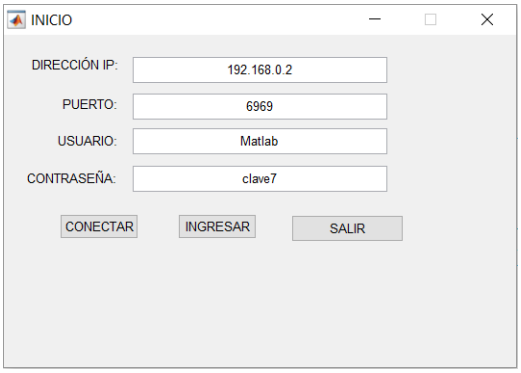


Figura 3.51: Ventana de inicio MATLAB

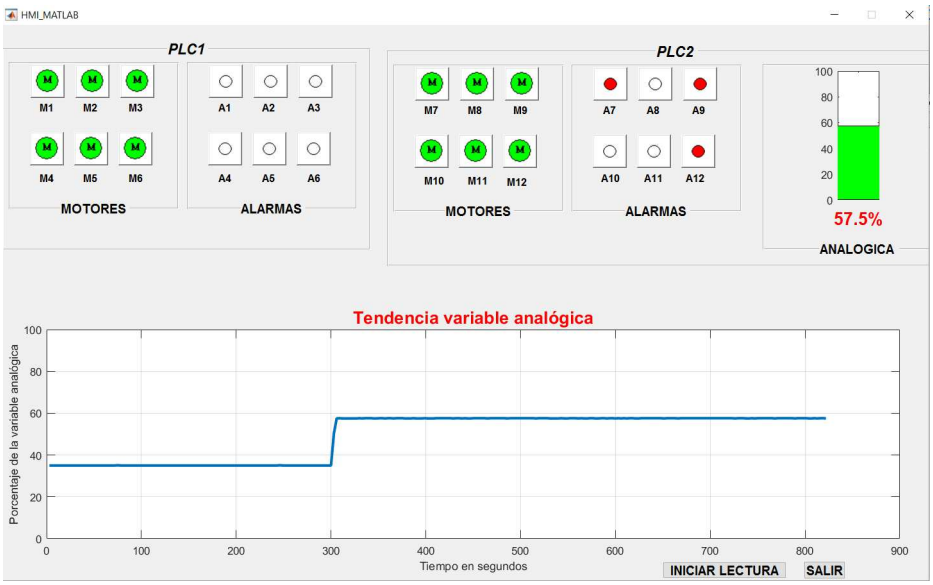


Figura 3.52: HMI desarrollado en Matlab

3.6. Funcionamiento del sistema

En las siguientes subsecciones se muestra los principales procesos que realizan los actores del sistema.

3.6.1. Autenticación

La Figura 3.53 muestra el diagrama de secuencia para la autenticación. La autenticación inicia cuando el usuario ejecuta la aplicación cliente, el servidor

debe ejecutarse con anterioridad para completar esta operación. Las acciones para realizar la autentificación con el servidor son:

- El usuario ingresa los datos en la ventana de inicio.
- El usuario presiona botón « conectar ». La aplicación cliente establece la conexión con el servidor.
- El usuario presiona el botón « aceptar ». La aplicación cliente envía los datos de autentificación al servidor. El servidor procesa la información y devuelve una confirmación, si el usuario es correcto ó incorrecto.
- La aplicación cliente al recibir la confirmación del servidor, procede a cerrar la ventana de inicio y muestra la interfaz gráfica SCADA correspondiente al usuario.

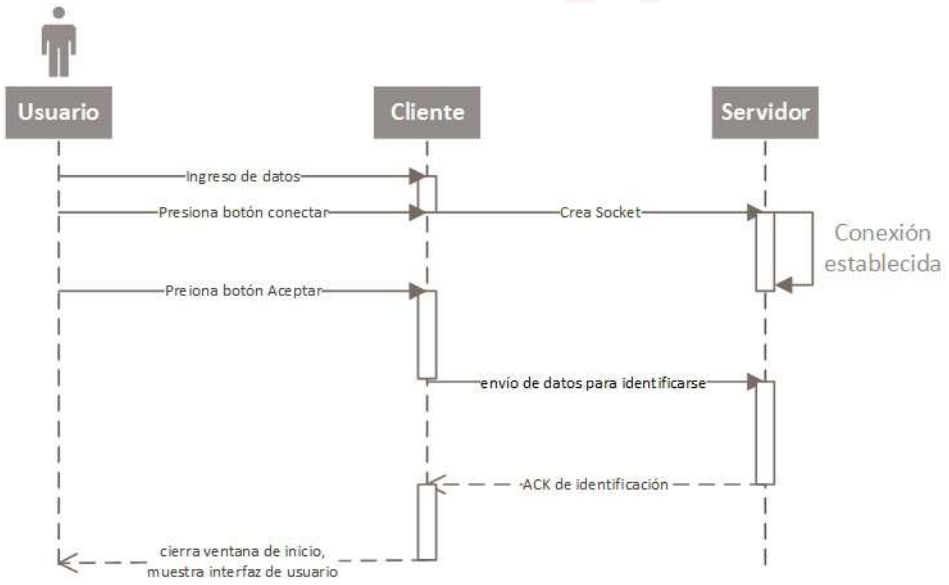


Figura 3.53: Diagrama de secuencia de autenticación

3.6.2. Interacción del sistema

La Figura 3.54 muestra el modelo de comunicación empleado entre el servidor (Python) y los clientes (Java y MATLAB). El servidor espera peticiones enviadas

desde los clientes mediante mensajes TCP, este mensaje incluye la acción que debe realizar.

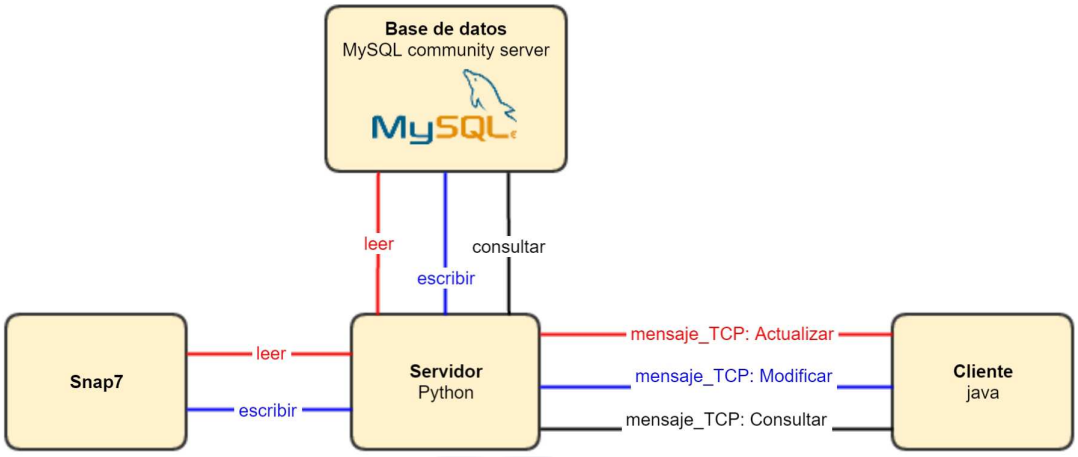


Figura 3.54: Comunicación utilizada en el sistema

Para cumplir estas peticiones, el servidor utiliza diferentes recursos. Las principales acciones que realiza el servidor son:

- **Actualizar:** El cliente realiza esta petición cuando requiere el estado de las variables monitoreadas. El servidor, mediante la librería Snap7, lee los estados de las variables del PLC. Los datos obtenidos son guardados en la base de datos y posteriormente enviados al cliente que realizó la solicitud.
- **Modificar:** Mediante esta petición, el cliente puede modificar los estados de cada variable de salida del PLC. El servidor modifica estas variables mediante la librería Snap7.
- **Consultar:** Esta consulta es realizada por el cliente para mostrar los datos históricos o generar un reporte en la aplicación. El servidor se conecta con MySQL y realiza las consultas que el cliente requiere.

3.6.3. Actualización de datos

La Figura 3.55 muestra el proceso de actualización de datos. Cuando el usuario se autentica en el servidor, activa un hilo que realiza la actualización de datos

después de tres segundos, este proceso es repetitivo. La aplicación cliente recibe los datos y los muestra en la interfaz gráfica.

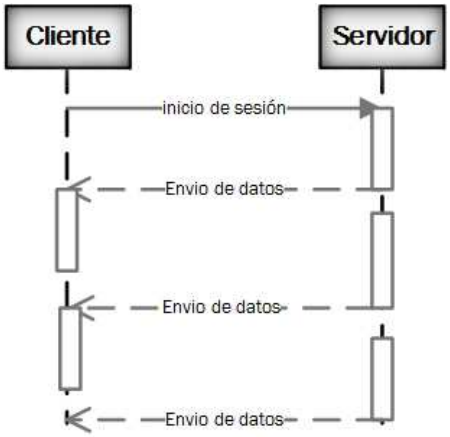


Figura 3.55: Proceso de actualización de datos

En el proceso de actualización de datos del cliente hace la comprobación del estado de las alarmas. Cuando una alarma se activa, ésta se agrega a la lista de alarmas activas de acuerdo a su prioridad.

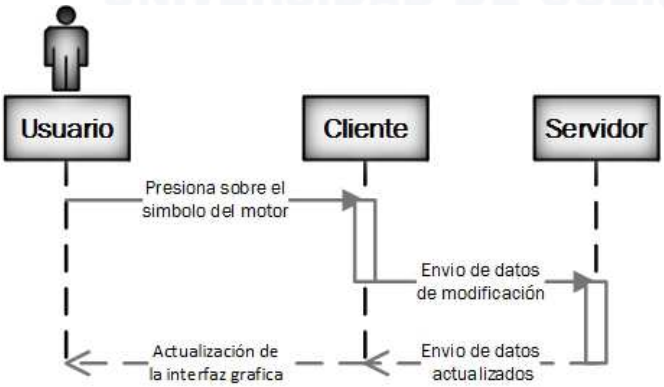


Figura 3.56: Proceso de cambio de estado de un motor

La Figura 3.56 muestra el proceso de cambio de estado de un motor. El usuario presiona el botón asociado al motor que desea encender ó apagar, la aplicación cliente realiza la identificación del motor y su estado actual, luego envía la petición al servidor. El servidor realiza el cambio de estado de la tag en el PLC correspondiente al motor.

Capítulo 4

Pruebas y análisis de los resultados del sistema

4.1. Análisis de la funcionalidad

El OSACIM fue expuesto a diferentes pruebas de funcionamiento e interacción con el usuario, dando periodos de tiempos de hasta 3 horas.

La Figura 4.1 muestra las funcionalidades que se ejecutaron durante las pruebas. La Tabla 4.1 muestra los resultados de las pruebas de funcionalidad, donde se denota ✓como correcto y χ como incorrecto. Obteniendo resultados óptimos de acuerdo al planteamiento del OSACIM como solución al problema.

Tabla 4.1: Resultados de las pruebas

FUNCIONALIDAD	RESULTADO
Inicio de sesión	✓
Cambio de estado de un motor	✓
Revisión de una alarma	✓
Gráfica de tendencias	✓
Generar reportes	✓
Visualización del historial de una alarma	✓
Interoperabilidad del sistema	✓

La interoperabilidad del sistema fue comprobada ejecutando al mismo tiempo los clientes de Java y MATLAB.

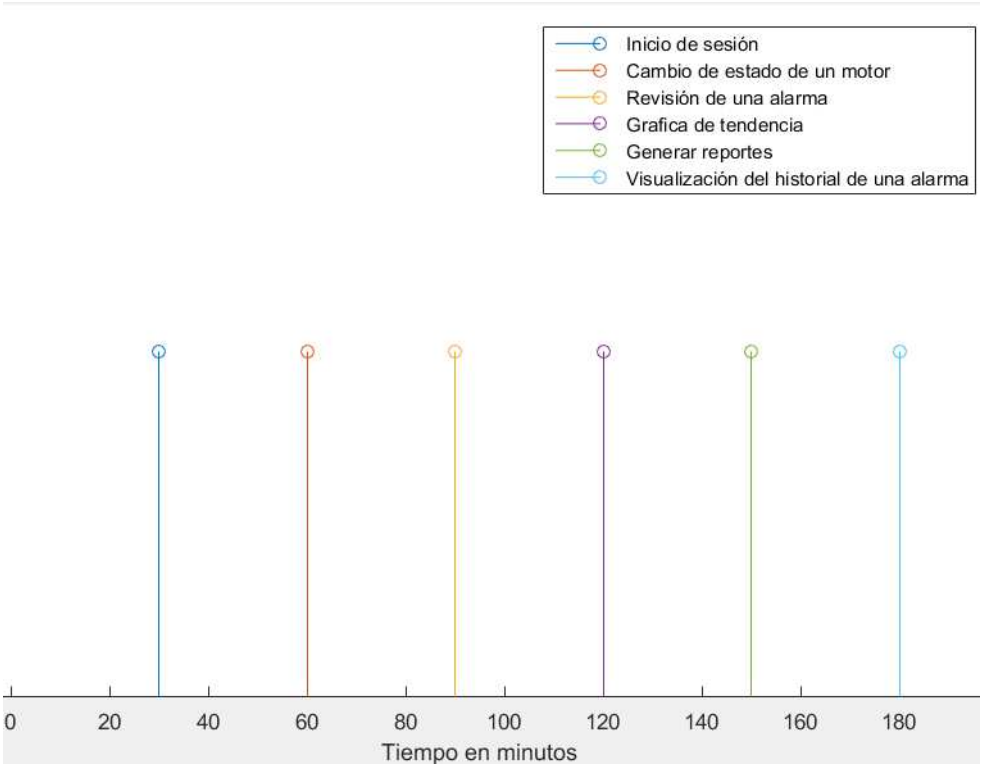


Figura 4.1: Línea de tiempo de ejecución de las pruebas de funcionamiento

4.2. Benchamarking con plataformas comerciales

SIMATIC en la actualidad es considerado el número uno mundialmente en el campo de la automatización en las diferentes industrias [31]. Esto se debe a que SIMATIC tiene las siguientes propiedades de un sistema integrado:

- **Ingeniería:** Máxima eficiencia durante las etapas del ciclo de vida de la máquina o instalación.
- **Comunicación**: Máxima transparencia de los datos a todos los niveles de automatización, basada en estándares probados.
- **Diagnóstico:** Minimización de los tiempos de parada con métodos de diagnósticos eficientes.

- **Seguridad:** Protección de personas y máquinas en el marco de un sistema global homogéneo e integrado.
- **Robustez:** Máxima aptitud para entornos industriales gracias a la gran robustez.
- **Tecnología:** Funciones tecnológicas integradas de contabilidad, medición, posicionamiento, regulación y control.
- **Alta disponibilidad:** Máxima disponibilidad con conceptos de redundancia homogéneas.

El costo de una solución comercial para una industria varía de acuerdo a las necesidades de planta. El número de tags a monitorizar es determinante en el costo; en SIMATIC WinCC el número de tags pueden ser 128, 512, 2048 ó 4096.

Para realizar una comparación entre el OSACIM desarrollado y SIMATIC WinCC se revisaron las características fundamentales de un sistema SCADA, descrito en los capítulos anteriores. La Tabla 4.2 muestra la comparación realizada en base a las características de SIMATIC WinCC y el OSACIM. La Tabla comparativa se basa en datos proporcionados por el manual de SIMATIC WinCC y las características de la solución propuesta en esta investigación.

Tabla 4.2: Tabla comparativa SIMATIC WinCC vs OSACIM

CARACTERÍSTICA	**SIMATIC WinCC**	**OSACIM**
Componentes de monitoreo y control.	Monitoreo y control de eventos y alarmas de acuerdo a la configuración de la interfaz.	Monitoreo de variables, alarmas, de acuerdo a la programación en el servidor.
Almacenamiento de valores medidos.	Realiza el almacenamiento de valores de acuerdo a la configuración sistema.	Almancenamiento de variables en DB personalizable, e.g. ante cambios de variables únicamente, de manera continua.
Sistema de Informes y reportes.	SIMATIC WinCC puede exportar la base de datos a un archivo de texto plano o a una aplicación comercial de acuerdo a la versión del programa.	Este dispone una interfaz gráfica para generar reportes en Excel o en una aplicación capaz de manipular archivos con extensión xls.
Opciones para la configuración del funcionamiento.	Posee varios paneles de configuración dinámicos, en las diferentes etapas de desarrollo del sistema SCADA.	La configuración del sistema se hace mediante la programación.
Arquitectura de base de datos.	Base de datos común para distintos dispositivos.	Base de datos para cada dispositivo.
Tipo de comunicación.	Posee una comunicación con OPC server.	Posee una comunicación basada sobre protocolo TCP/IP.
Administración de usuarios y protección de acceso.	Posee un sistema de control de acceso asociados grupos de usuarios, contraseñas, autorizaciones.	Administración de usuarios centralizada para todo el sistema.
Sistemas Operativos.	▪ Windows XP Professional (32 bits) SP2/SP3. ▪ Windows XP Embedded. ▪ Windows Vista Business (32 bits +SP1). ▪ Windows Vista Ultimate (32 bits +SP1).	La aplicación es multiplataforma puede ejecutarse en Windows de 32 y 64 bits, Linux, y otros sistemas operativos que permitan la ejecución de Java y Python.
Tipo de interfaz gráfica.	Posee una interfaz dinámica con gráficos propios ó importando de otras aplicaciones.	La interfaz gráfica es limitada y se desarrolla de acuerdo a cada proceso a simular.

4.3. Análisis del despliegue de la función de calidad del sistema

El despliegue de la función de calidad (QFD, por sus siglas en inglés) permite identificar los requerimientos que tienen los clientes para satisfacer alguna necesidad. Las necesidades son traducidas en características técnicas, para así desarrollar un producto que satisfaga estas peticiones.

Fue desarrollado en Japón, en la década de los 60. Los principales objetivos del QFD son [32]:

- Establecer calidad tanto en el diseño como en la planificación.
- Realizar *benchmarking* con productos de otras marcas.
- Desarrollar productos competitivos dentro del mercado.
- Reducir tiempo y costo en el desarrollo de un producto.

La herramienta basada en QFD se denomina casa de la calidad (HOQ, por sus siglas en inglés). Esta técnica permite realizar un *benchmarking* entre el software desarrollado y uno de la competencia.

Para desarrollar la HOQ, se aplicaron encuestas a diferentes profesionales conocedores de las necesidades de un sistema SCADA. Las encuestas se realizaron en el software de encuesta en línea gratuito SurveyMonkey.

El modelo de encuesta se muestra en el Apendice D. Los resultados obtenidos se muestran en el Apendice E. Estos resultados pertenecen a 31 encuestas realizadas.

En base a las encuestas, las necesidades más importantes de un sistema SCADA se muestran en la Tabla 4.3. La importancia se califica a través de un valor numérico del 1 al 5, siendo 5 lo más alto y 1 lo más bajo.

Tabla 4.3: Necesidades de un sistema SCADA

Número	**Necesidad**	**Importancia**
1	Seguridad de los datos	4
2	Protección de acceso de cada usuario mediante autenticación	5
3	Interfaz gráfica animada	3
4	Compatibilidad con software de otras marcas	3
5	Avisos al operario ante una falla en el sistema	5
6	Fácil de manejar	4
7	Escalabilidad de software y hardware	3
8	Precio accesible	4
9	Manejo de una base de datos	5
10	Servidor centralizado para todos los dispositivos	4
11	Multiplataforma	4
12	Exportar los datos en Excel	4
13	Almacenamiento,de los datos en intervalos específicos de tiempo	5
14	Diferentes búsquedas dentro de la base de datos	3
15	Actualización y monitoreo de los datos en intervalos específicos de tiempo	4
16	Fácil de instalar	1

Los requerimientos técnicos (RT) se desarrollaron en base a las necesidades de los usuarios. La HOQ diseñada para el sistema se muestra en el Apéndice G. La prioridad técnica de cada RT, mostrada en el Apéndice G, nos indica la contribución numérica que cada RT aporta para cada necesidad del cliente.

Los resultados obtenidos del *benchmarking* entre el OSACIM y el software WinCC se muestra en la Tabla 4.4. La meta planeada indica los puntos en los cuales el

software puede mejorar y así brindar un mejor servicio a los requerimientos del cliente.

Tabla 4.4: Resultados del benchmarking con WinCC

Necesidad	Aplicación OSACIM	Competidor: WinCC	Meta planteada
1^o	3	5	4
2^o	5	5	5
3^o	3	4	4
4^o	4	4	4
5^o	5	5	5
6^o	4	4	4
7^o	4	5	5
8^o	5	3	5
9^o	4	5	5
10^o	2	5	4
11^o	4	1	4
12^o	5	5	5
13^o	3	5	4
14^o	4	4	5
15^o	3	5	4
16^o	5	4	5

Los valores mostrados indican el nivel que satisface el sistema en base a las necesidades de los clientes.

4.4. Limitaciones

Las limitaciones del sistema son las siguientes:

- Los PLCs compatibles con la aplicación son de la marca Siemens y modelos S7-200, S7-300, S7-400, S7-1200 y S7-1500. Todos estos modelos utilizan la comunicación Ethernet.
- El número de clientes que pueden conectarse simultáneamente al servidor está limitado a cinco.
- La generación de reportes en un cliente está habilitada únicamente cuando no hay un procesamiento de reporte en la aplicación servidor.

- El programa no permite generar múltiples reportes, procesados por un cliente. La generación de reportes vuelve a habilitarse cuando el servidor termina el procesamiento de la petición anterior.
- El sistema está diseñado en base a peticiones. El tiempo máximo para la actualización de datos en el SCADA es de tres segundos. Esto se debe a los procesos que realiza el servidor antes de responder esta petición.

Capítulo 5

Conclusiones

5.1. Conclusiones

Los sistemas SCADA son muy importantes dentro de una empresa, ya que proporcionan un control y monitoreo de las variables de proceso, ayudando a la producción y competencia dentro del mercado.

Los componentes del software están desarrollados en lenguaje de código abierto, permitiendo a los usuarios ejecutar el sistema SCADA en cualquier plataforma. La ejecución del software en una plataforma distinta a Windows, por ejemplo Linux, permite mejorar la seguridad e interoperabilidad del software.

Para acceder a un sistema open-source completo, es decir hasta el nivel de supervisión del modelo CIM, se necesita implementar hardware de arquitectura libre. Esto permite liberar a las empresas de proveedores de marcas específicas. El OSACIM desarrollado se puede acoplar fácilmente a hardware libre, por ejemplo, Arduino, y utilizarse como solución para empresas pequeñas, donde el ruido no sea un factor que afecte al funcionamiento.

El OSACIM desarrollado puede ejecutarse paralelamente con un software comercial utilizado en la industria. Por esta razón, es independiente y no invasivo cuando existen soluciones de automatización previamente instaladas.

El OSACIM ofrece un ahorro en el costo de licencias, estos recursos pueden ser asignados a otras necesidades en la industria.

El estudio de calidad (QFD) permitió determinar que la aplicación desarrollada solventa las necesidades más importantes de un SCADA. Nuevos requerimientos pueden resolverse mediante trabajos futuros sobre la aplicación desarrollada, por ejemplo, la implementación de una comunicación con el protocolo OPC.

La comunicación basada en *sockets* ofrece una interoperabilidad con programas comerciales, siendo una buena alternativa a la comunicación OPC. Esta comunicación podría dar soporte a programas para análisis de datos y optimización del sistema, es decir escalar un nivel en el modelo CIM.

Los reportes son generados en el servidor para evitar la sobrecarga de flujo de datos en la comunicación TCP/IP. Una solución a este problema comprendería el desarrollo de un servidor de reportes.

5.2. Trabajos futuros

El OSACIM implementado se puede mejorar optimizando algunas funciones de las aplicaciones en el cliente y en el servidor. A continuación se detallan algunos de los trabajos futuros:

- Implementar un servidor de reportes con JasperReports para los clientes Java. Al desarrollar un servidor de reportes se retirará la sobrecarga de procesamiento de la aplicación servidor, proporcionando al sistema mejor funcionalidad a la hora de generar un reporte o realizar una consulta.
- Sustituir la conexión TCP/IP por una conexión OPC. Esto ayudaría a interconectar con mayor facilidad los PLCs de SIEMENS con otros dispositivos.
- Implementar la comunicación entre el cliente y servidor mediante Web Services, Java-RMI, o CORBA. Éstas opciones proporcionarían una mayor escalabilidad al sistema, permitiendo el acceso de forma remota o desde un dispositivo móvil.

- Generalizar el número de conexiones del servidor con los clientes. Esto se puede lograr mediante el cambio de lenguaje de programación del servidor. El servidor en Python permite conectar cinco clientes como máximo.
- Implementar una capa de seguridad de sockets (SSL, por sus siglas en inglés) o seguridad en la capa de trasporte (TLS, por sus siglas en inglés) para mejorar la seguridad del sistema. Las aplicaciones del servidor y cliente poseen características básicas de seguridad.

Apéndices

Apéndice A

Requisitos previos para la ejecución del sistema

A.1. Requisitos para instalar el servidor

Los programas necesarios para que el servidor se ejecute correctamente son:

- El gestor de base de datos MySQL Community Server 6.3. Para crear las tablas se utiliza la extensión MySQL Workbench 6.3 CE. Esta aplicación viene por defecto en el paquete del servidor MySQL.
- Python 3.5 para la programación del software. El ambiente de desarrollo integrado (IDE, por sus siglas en inglés) utilizado para este lenguaje de programación es Ninja IDE.
- El software Microsoft Excel, para abrir los reportes generados en el servidor.

Los paquetes que se necesitan instalar en Python para ejecutar el servidor son:

- El conector MySQL-Python para la conexión con el servidor de base de datos.
- La librería Snap7 0.4 para la comunicación con los dispositivos de campo. También se debe copiar los archivos .dll y .lib de Snap7 en la carpeta win32 en la PC que se ejecuta el servidor.

- La librería Crypto para la encriptación de los mensajes TCP.
- La librería xlwt para la creación de archivos Excel.

A.2. Requisitos para instalar el cliente Java

Los programas necesarios para que el cliente se ejecute correctamente son:

- Java con JDK.
- El IDE Netbeans para la programación.

Las librerías que se requieren instalar para que funcione el software SCADA son:

- **commons-codec-1.10.jar:** Sirve para encriptar los menjases TCP.
- **jcalendar-1.4.jar:** Permite ingresar un calendario en la interfaz para realizar las consultas por fechas.
- **jcommon-1.0.23.jar:** Utilizada para graficar las tendencias.
- **jfreechart-1.0.19.jar:** Utilizada para graficar las tendencias.
- **jfreechart-1.0.19-experimental.jar:** Utilizada para graficar las tendencias.
- **jfreechart-1.0.19-swt.jar:** Utilizada para graficar las tendencias.
- **jl1.0.1.jar:** Se utiliza para insertar audio en Java. En el OSACIM se utiliza para reproducir las alarmas generadas por diferentes fallas en el sistema.

Apéndice B

Diagrama de clases del sistema

B.1. Diagrama de clases del servidor

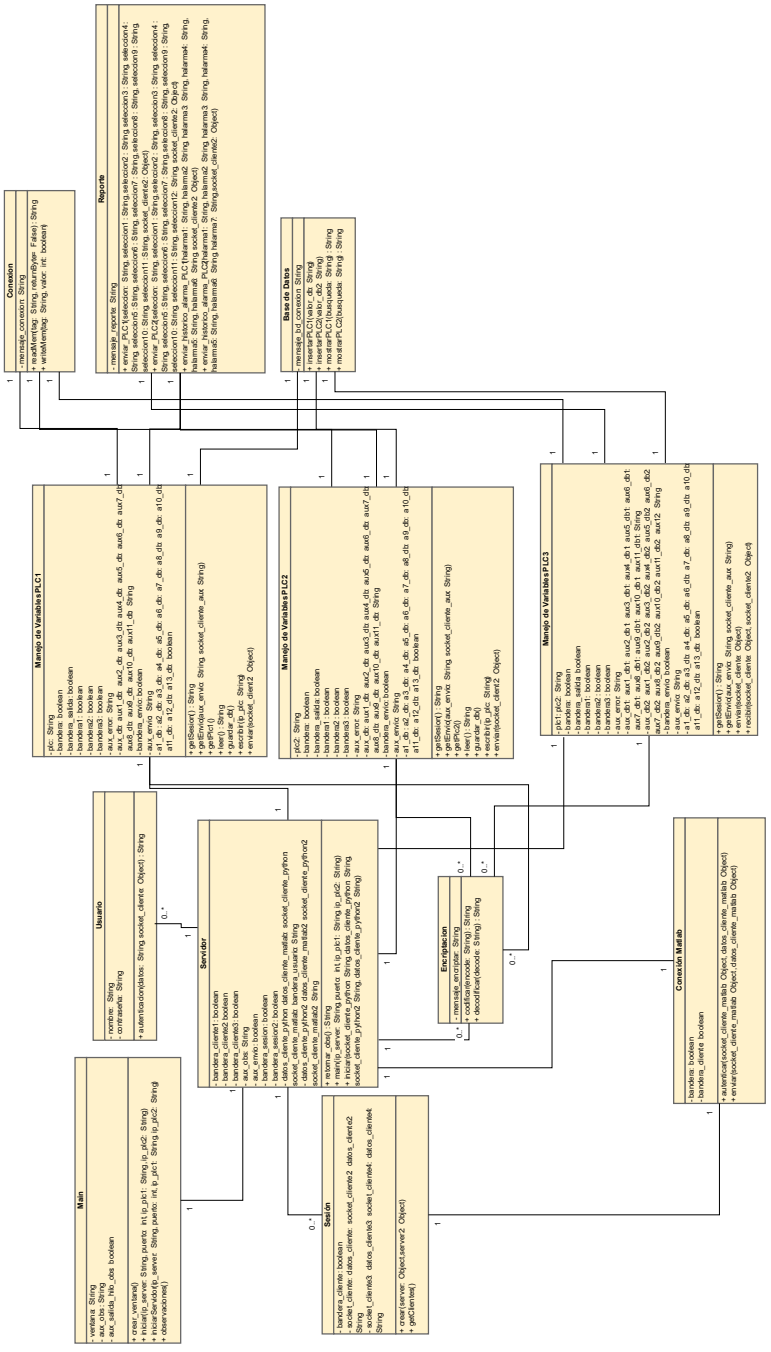


Figura B.1: Diagrama de clases del servidor

B.2. Diagrama de clases del cliente Java

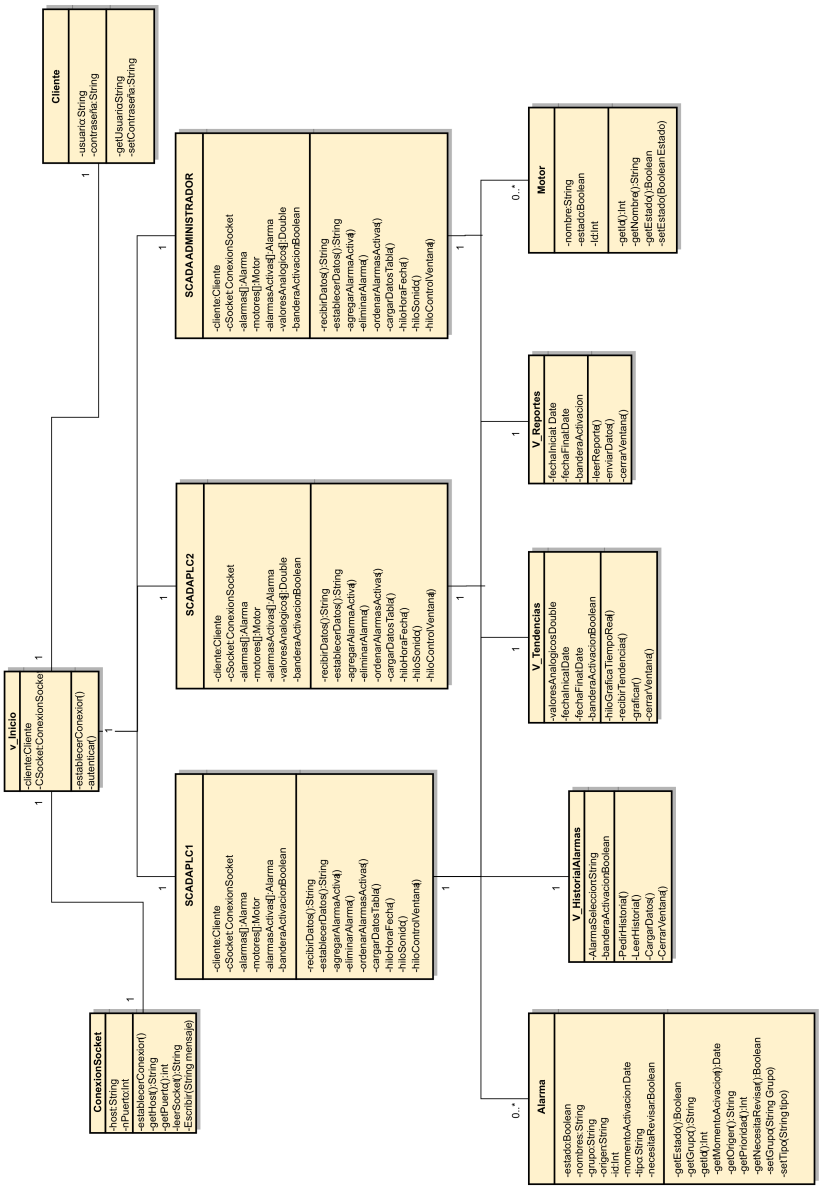


Figura B.2: Diagrama de clases del cliente Java

Apéndice C

Manual de usuario del sistema

En las siguientes subsecciones se muestra los pasos que se siguen para ejecutar y utilizar las funciones disponibles en el sistema.

C.1. Servidor

La Figura C.1 muestra la interfaz de inicio del servidor, en esta interfaz se configura los siguientes parámetros:

- La dirección IP del servidor.
- Las direcciones IP de los PLCs
- El puerto por el cual se enviarán los datos.
- El modelo del PLC, el sistema esta implementado para que trabaje con los PLCs S7-1200 de Siemens.

Cuando todos los parámetros de conexión estén configurados, se procede a iniciar el sistema presionando el botón « conectar » (Ver la Figura C.2).

Cuando el servidor este ejecutándose correctamente se muestra un mensaje en la interfaz de inicio (ver Figura C.3).

Figura C.1: Interfaz de inicio del servidor

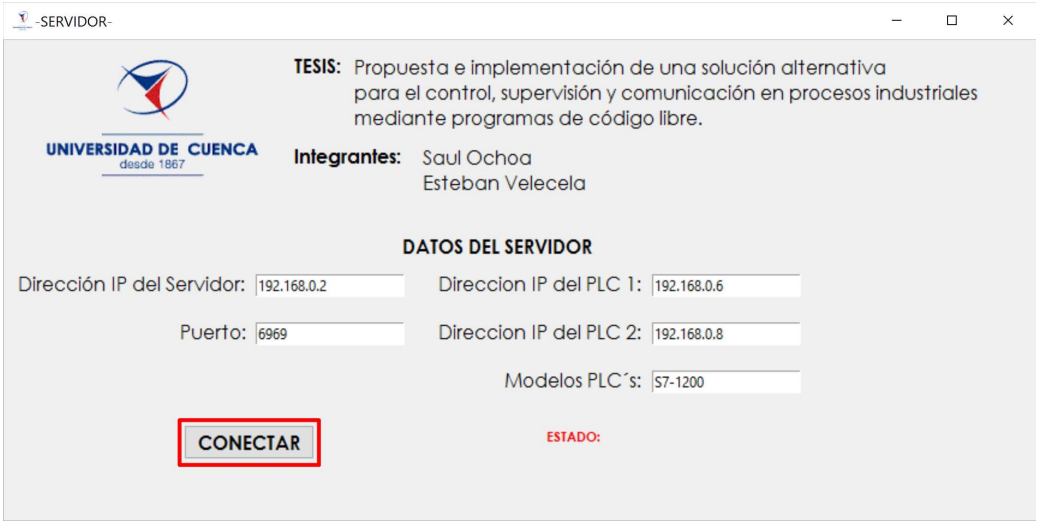


Figura C.2: Inicio del servidor

Figura C.3: Ejecución correcta del servidor

C.1.1. Reporte de problemas en el servidor

La información del estado del servidor es muy importante para el usuario, el sistema provee los errores generados en el software mediante mensajes. Éstos son:

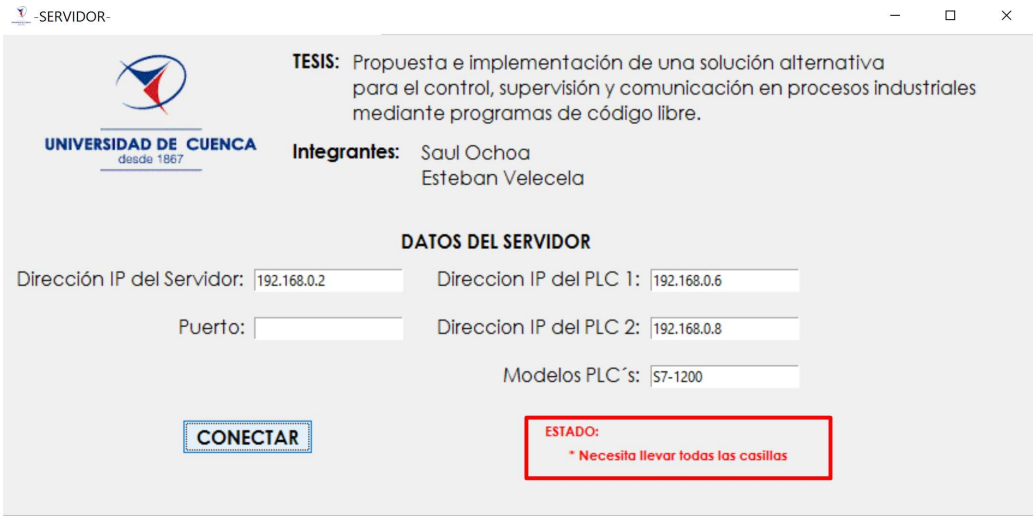


Figura C.4: Error en el inicio del servidor por falta de información

Falta de información para el inicio del servidor

Si existe falta de información para el inicio del servidor, esto se muestra en la interfaz del servidor (ver Figura C.4).

Dirección IP del servidor incorrecta

La Figura C.6 muestra en mensaje presentado en la interfaz cuando existe un error al ingresar la dirección IP del servidor.

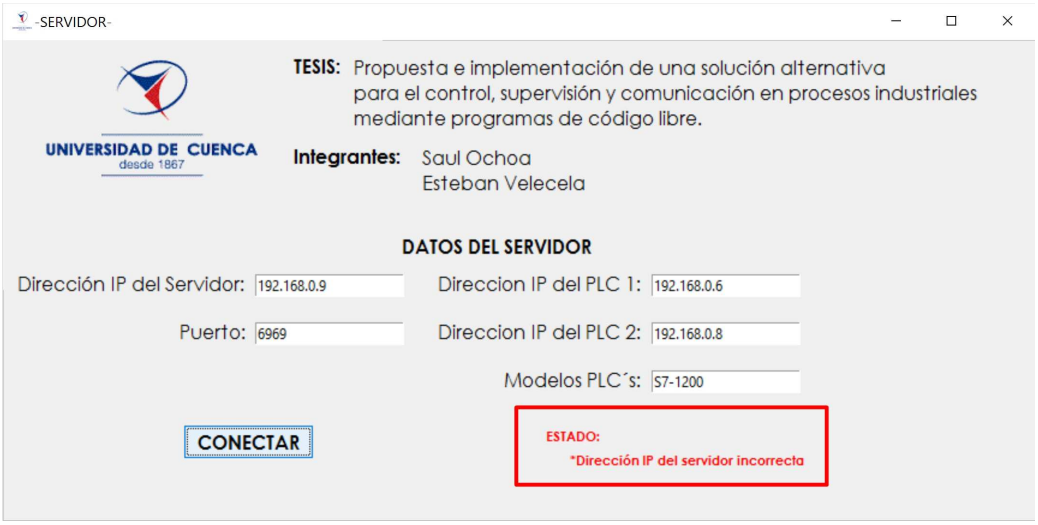


Figura C.5: Error generado por la dirección IP incorrecta del servidor

Falla en la conexión con el cliente

Si existe una desconexión de la red del cliente SCADA, el servidor informada a su operario mediante un mensaje en la interfaz (ver Figura C.6). Para iniciar una conexión con el cliente, se pide al usuario que ejecute de nuevo el servidor.

Los reportes son generados en el servidor, de acuerdo al tipo de búsqueda que realice el cliente. El archivo Excel generado se abre automáticamente cuando la consulta en la base de datos termine (ver Figura C.7).

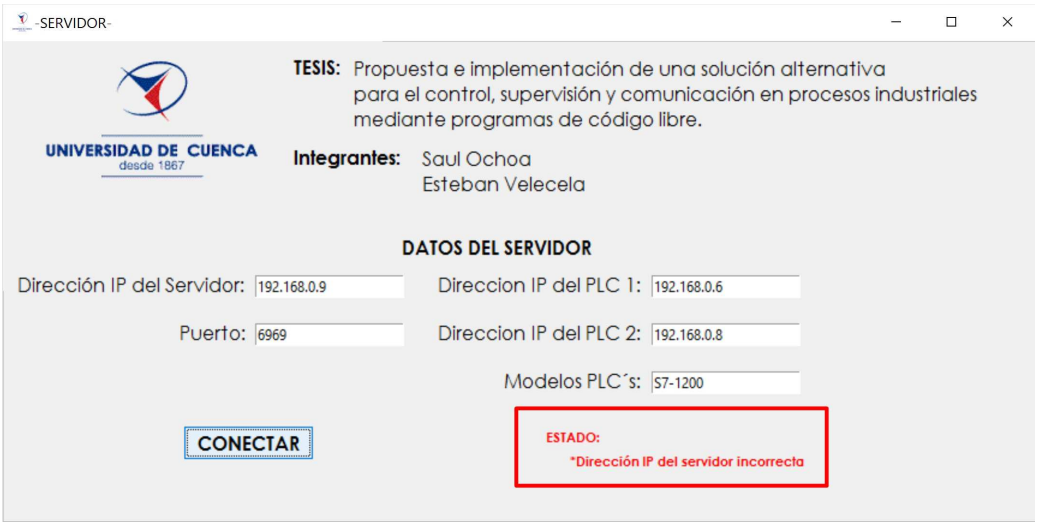


Figura C.6: Error producido por la pérdida de conexión con el cliente

Figura C.7: Ejecución automática del archivo que contiene la consulta en el servidor

C.2. Cliente Java

C.2.1. Proceso de autenticación

La Figura C.8 muestra la ventana de autenticación, en esta se configuran los siguientes parámetros:

- La dirección IP del servidor.
- El número del puerto de la aplicación servidor.
- Usuario.
- Contraseña.

El proceso de autenticación se describe a continuación:

1. Configuración de los parámetros de conexión y autenticación descritos antes.
2. Presionar el botón « conectar » (recuadro rojo, Figura C.8).
3. Presionar el botón « aceptar »(recuadro naranja, Figura C.8).

Figura C.8: Ventana de autenticación

En el proceso de autenticación puede provocarse un error generado por el usuario.

La Figura C.9 muestra el error provocado por el usuario al presionar el botón « aceptar » antes de presionar el botón « conectar ». El mensaje se cierra cuando presionamos el botón « aceptar ».

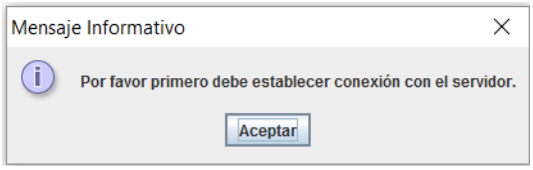


Figura C.9: Error provocado por el usuario en el proceso de autenticación

C.2.2. Cambio de estado de los motores

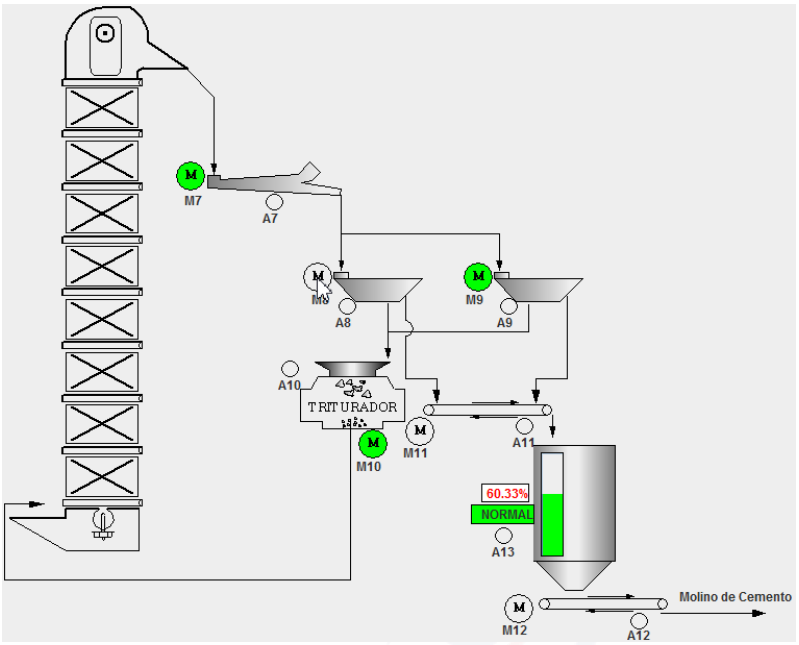
El usuario debe colocarse con el puntero del ratón sobre el motor que desea cambiar de estado, al hacer clic sobre el motor este cambiará de estado.

La Figura C.10 muestra el proceso para el cambio de estado en un motor. La Figura C.2.2 muestra como es el posicionamiento del ratón, mientras que, la Figura C.2.2 muestra el cambio de estado después de hacer clic.

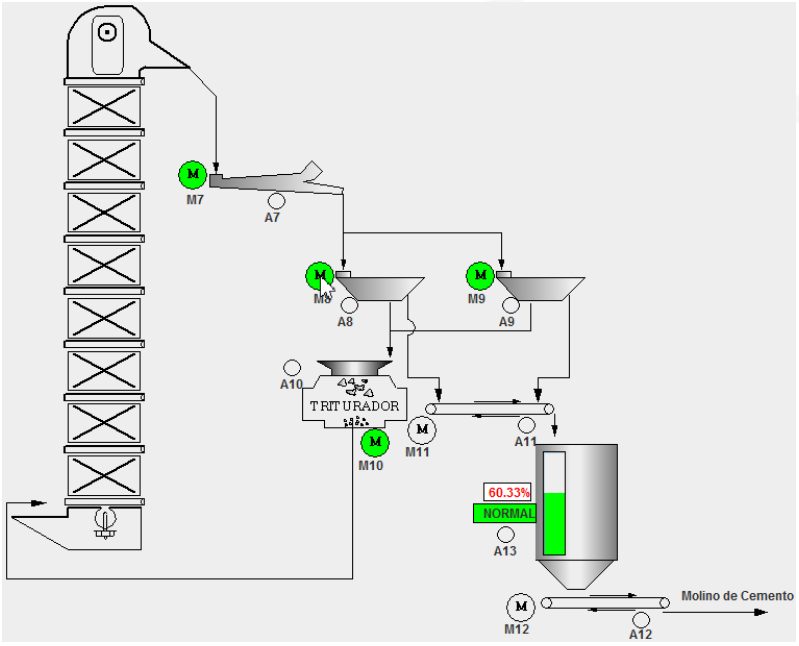
C.2.3. Revisión de alarmas

La revisión de una alarma generada se realiza en la interfaz principal del usuario, cuando una alarma se genera se muestra los siguientes parámetros de la alarma:

- Nombre
- Fecha
- Hora
- Origen
- Prioridad
- PLC
- Tipo
- Estado de revisión



(a) Posicionamiento del puntero del ratón



(b) Clic del ratón

Figura C.10: Cambio de estado del motor M8

El usuario debe hacer clic sobre la columna de estado de revisión para realizar la revisión de la alarma. Cuando una alarma es revisada el contador de alarmas disminuye.

La Figura C.11 muestra en un rectángulo rojo la casilla donde el usuario debe hacer clic para realizar la revisión de una alarma, en el rectángulo azul se encuentra el contador de alarmas de la interfaz.

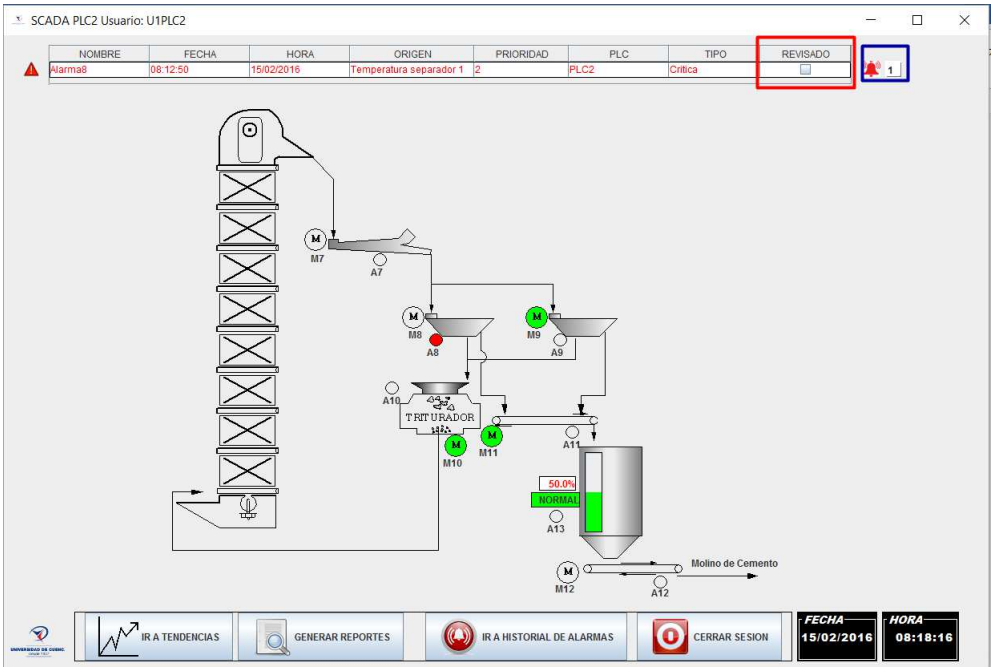


Figura C.11: Columna de estado de revisión y contador de alarmas

C.2.4. Proceso de visualización de tendencias

El proceso de visualización de tendencias está disponible en las interfaces de los usuarios U1PLC2, U2PLC2, U3 y U3lock. La implantación para graficar tendencias se hizo de dos formas:

- Tendencia a tiempo real.
- Tendencia de un intervalo de tiempo.

El usuario debe hacer clic en el botón « ir a tendencias » ubicado en la barra de menú de navegación.

La Figura C.12 muestra el botón « ir a tendencias » en la interfaz del usuario U1PLC2 correspondiente al PLC2.

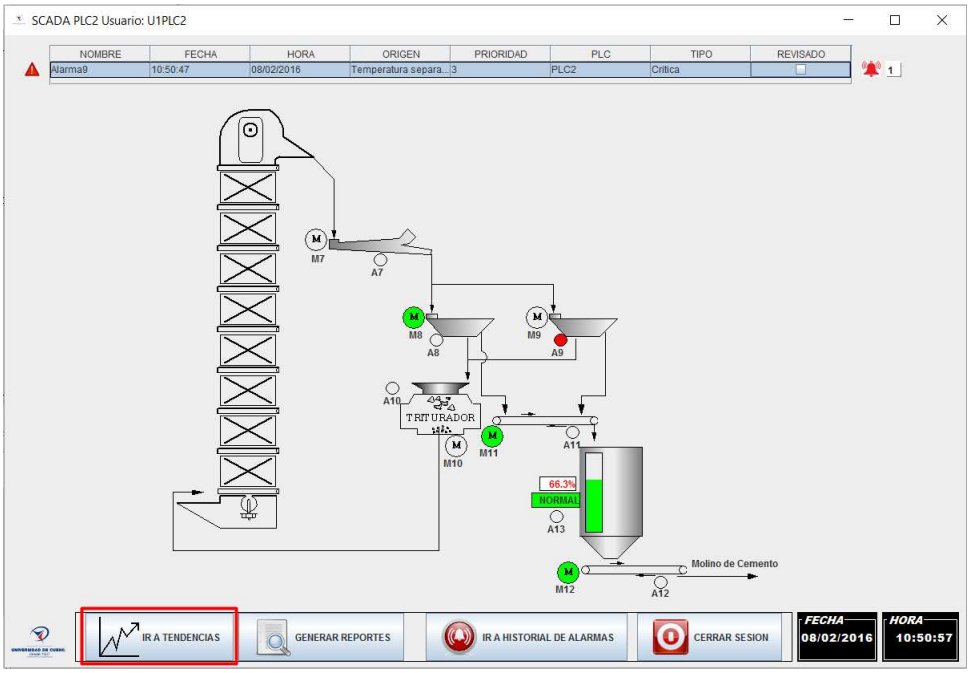


Figura C.12: Botón para ir a las tendencias en la interfaz del PLC2

Tendencia a tiempo real

La tendencia de tiempo real se ejecuta por defecto cuando la ventana de tendencias está abierta. La variable graficada corresponde a la variación de la entrada analógica actualizada cada 3 segundos. En la pestaña de tendencia tiene un botón para controlar la actualización de la variable.

La Figura C.13 muestra la ventana de tendencias, donde se puede ver el botón « Play/Pause » para el control de la actualización de datos de la gráfica.

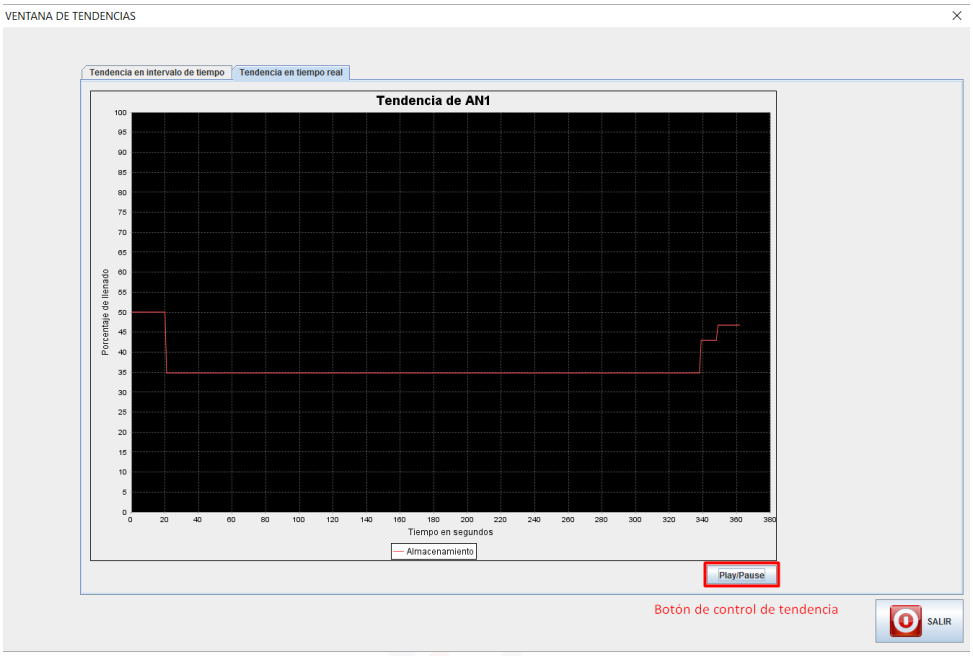


Figura C.13: Pestaña de tendencia de tiempo real

Tendencia en intervalo de tiempo

Para realizar la gráfica de tendencia en un intervalo de tiempo, se ejecuta los siguientes pasos:

- Presionar el botón « ir a tendencia ».
- Seleccionar la pestaña tendencia en intervalo de tiempo.
- Configurar el intervalo de tiempo en el panel de configuración.
- Presionar el botón « aceptar » del panel de configuración.

La Figura C.14 muestra el panel de configuración de la pestaña de tendencia en un intervalo de tiempo.

Figura C.14: Panel de configuración de la ventana de tendencias

C.2.5. Proceso de generación de reportes

El usuario puede generar cuatro tipos de reportes, éstos son:

- Reporte por fecha.
- Reporte de estado de variables digitales.
- Reporte de alarmas revisadas.
- Reporte de toda la base de datos.

El usuario debe hacer clic en el botón « generar reportes » de la barra de menú de navegación. La Figura C.15 muestra el botón para generar reportes en la interfaz del usuario.

Para generar un reporte el usuario debe seguir los siguientes pasos:

- Seleccionar el tipo de reporte.
- Configurar la fecha y la hora de inicio.

- Configurar la fecha y la hora final.
- Presionar el botón « generar reporte ».

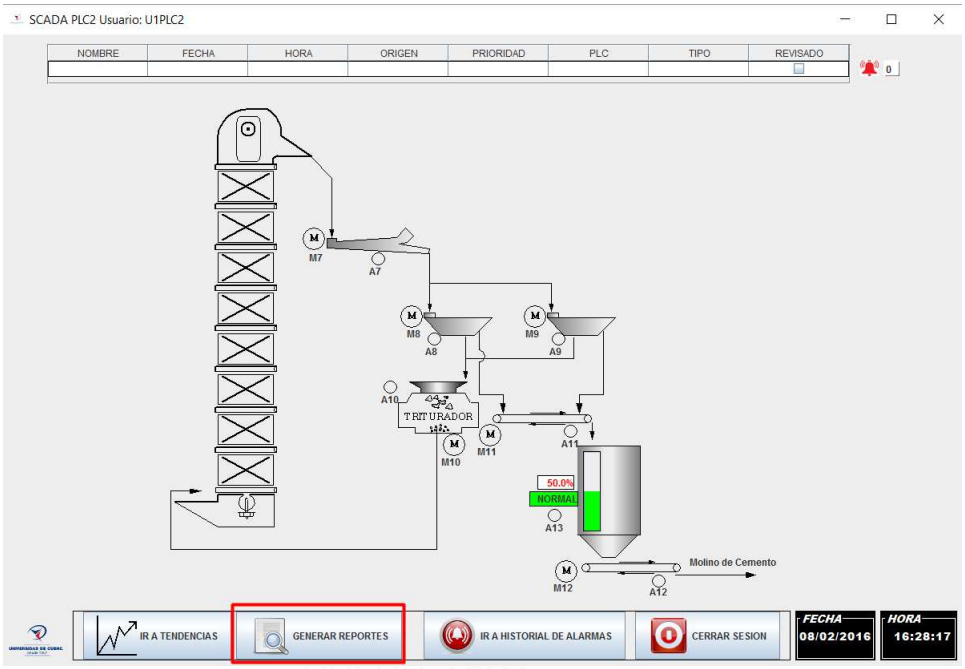


Figura C.15: Botón para generar reportes en la interfaz gráfica del PLC2

La Figura C.16 muestra un ejemplo de configuración para generar un reporte por fecha. Cuando se selecciona un reporte completo se deshabilita la configuración de datos iniciales y finales.

C.2.6. Proceso para visualizar historial de alarmas

El usuario debe hacer clic en el botón « ir a historial de alarmas » de la barra de menú de navegación, ejecutándose la ventana de historial de alarmas.

La Figura C.17 muestra la ubicación de botón en la barra de menú de navegación de la interfaz de usuario.

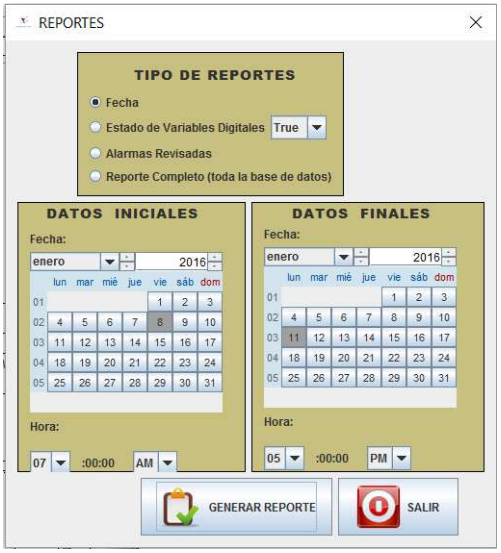


Figura C.16: Ejemplo de configuración de un reporte

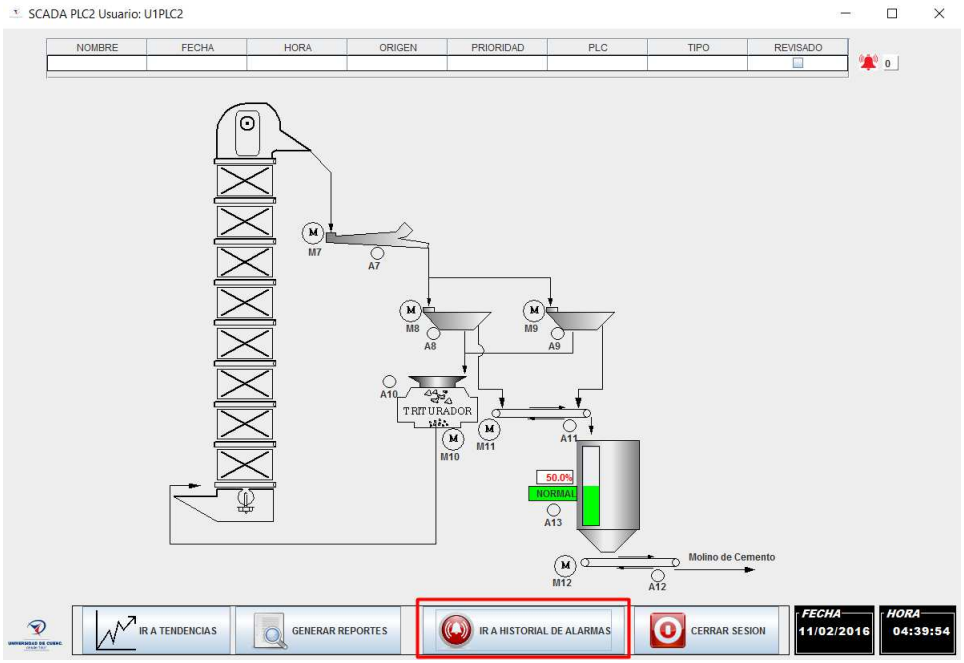


Figura C.17: Botón « ir a historial de alarmas » en la barra de menú de navegación

El usuario debe realizar los siguientes pasos para hacer una consulta del historial de una alarma:

- Seleccionar la alarma que se desea consultar su historial.
- Hacer clic en el botón « pedir historial ».

La Figura C.18 muestra un ejemplo de visualización de un historial de una alarma.

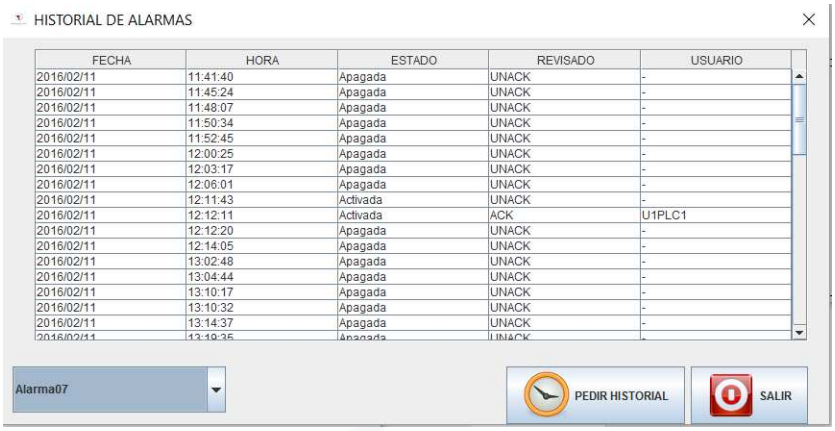


Figura C.18: Ejemplo de historial de una alarma

C.3. Cliente MATLAB

El cliente MATLAB simula el funcionamiento de un HMI. Al ejecutar la aplicación en MATLAB se muestra la ventana principal donde se realiza la autenticación con el servidor. El usuario debe llenar los siguientes parámetros:

- Dirección IP del servidor
- Número de puerto de la aplicación cliente
- Usuario
- Contraseña

La Figura C.19 muestra la ventana de inicio de la aplicación cliente en MATLAB.

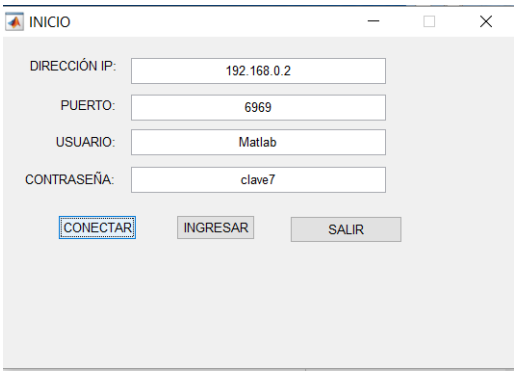


Figura C.19: Ventana de inicio de sesión del cliente MATLAB

C.3.1. Proceso de autenticación

Para realizar la autenticación, el usuario debe seguir los siguientes pasos:

- Llenar correctamente los datos en la ventana principal.
- Establecer la conexión con el servidor presionando el botón « conectar ».
- Realizar la autenticación presionando el botón « ingresar ».

Cuando el usuario se ha identificado correctamente con el servidor, se cierra la ventana principal y muestra la interfaz HMI.

La Figura C.20 Muestra la interfaz HMI de MATLAB. La interfaz HMI muestra los estados actuales de las variables del sistema.

C.3.2. Cambio de estado de motores

La interfaz HMI de MATLAB funciona de forma similar a la del cliente Java, para realizar un cambio de estado de un motor, el usuario debe posicionar el puntero sobre el motor y realizar un clic.

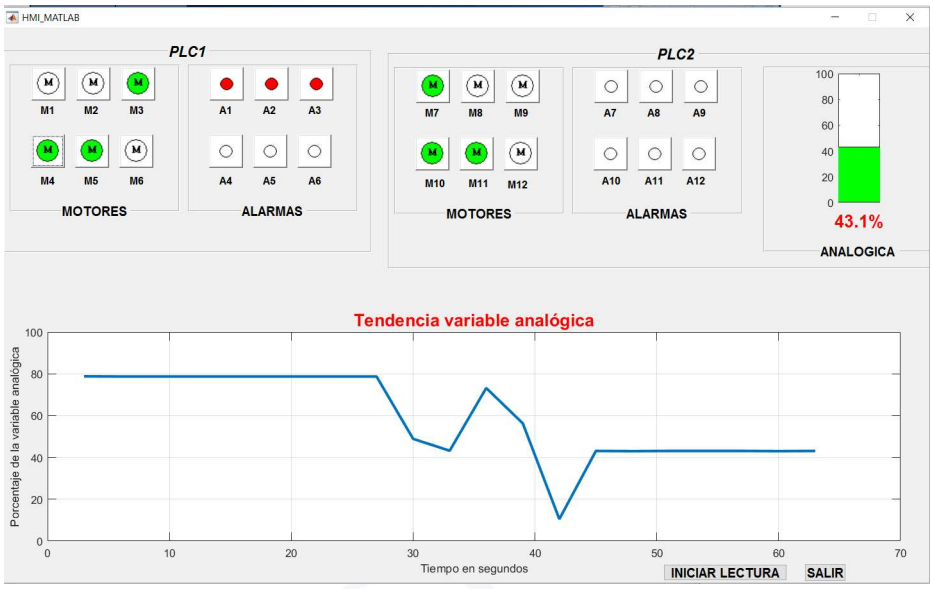
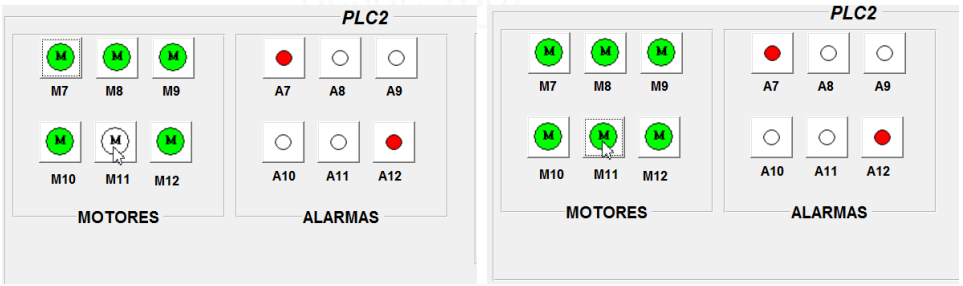


Figura C.20: Ventana de inicio del cliente MATLAB

La Figura C.21 muestra el proceso para el cambio de estado en un motor. La Figura C.3.2 muestra como es el posicionamiento del ratón, mientras que, la Figura C.3.2 muestra el cambio de estado después de hacer clic.



(a) Posicionamiento del puntero del ratón (b) Clic del ratón

Figura C.21: Cambio de estado del motor en MATLAB

Apéndice D

Modelo de encuesta realizada

1. ¿Cuáles considera usted que son los requerimientos más importantes en un sistema SCADA industrial? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)

	1	2	3	4	5
a. Seguridad de los datos	◯	◯	◯	◯	◯
b. Protección de acceso mediante autenticación	◯	◯	◯	◯	◯
c. Generación de reportes e históricos	◯	◯	◯	◯	◯
d. Interfaz gráfica dinámica	◯	◯	◯	◯	◯
e. Compatibilidad con softwares de otras marcas	◯	◯	◯	◯	◯
f. Avisos al operario ante una falla en el sistema	◯	◯	◯	◯	◯
g. Fácil de manejar	◯	◯	◯	◯	◯
h. Escalable	◯	◯	◯	◯	◯
i. Precio accesible	◯	◯	◯	◯	◯
j. Manejo de una base de datos	◯	◯	◯	◯	◯
k. Otro (especifique)	----------				

2. Cual considera usted que es la mejor arquitectura para la base de datos de un sistema SCADA

◯ Servidor centralizado para todos los dispositivos

◯ Base de datos descentralizada para cada dispositivo

3. Según su experiencia, ¿Cree usted importante que los sistemas SCADA sean multiplaforma, es decir funcionen en varios sistemas ope-

rativos? (Asigne un numero número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)

4. En que aplicación considera usted que se deben exportan los datos para generar un reporte.

- ◯ Microsoft Word (.doc)
- ◯ Excel (.xls)
- ◯ Adobe Reader (.pdf)
- ◯ Access (.accdb)
- ◯ Block de notas (.txt)
- ◯ Otro (especifique)

  _____

5. ¿Cuál considera usted que son las mejores búsquedas, dentro de la base de datos, para generar un reporte? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)

	1	2	3	4	5
a. Rango de fecha	◯	◯	◯	◯	◯
b. Estado de la variable	◯	◯	◯	◯	◯
c. Alarmas revisadas	◯	◯	◯	◯	◯
d. Toda la base de datos	◯	◯	◯	◯	◯

6. ¿Qué tan importante considera usted generar gráficas de tendencias de una variable? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)

7. ¿Cuál considera usted que es la mejor administración de usuarios y protección de acceso?

◯ Grupos de usuarios

◯ Usuario centralizado para todo el sistema

8. ¿Cuál considera usted que sería el tiempo óptimo de respuesta en segundos del sistema al actualizar el estado de las variables?

---

9. ¿Cuál considera usted que es la mejor manera de almacenar los datos en el servidor de base de datos?

◯ En intervalos específicos de tiempo

◯ Por cada cambio en la(s) variable(s) que se monitorea(n)

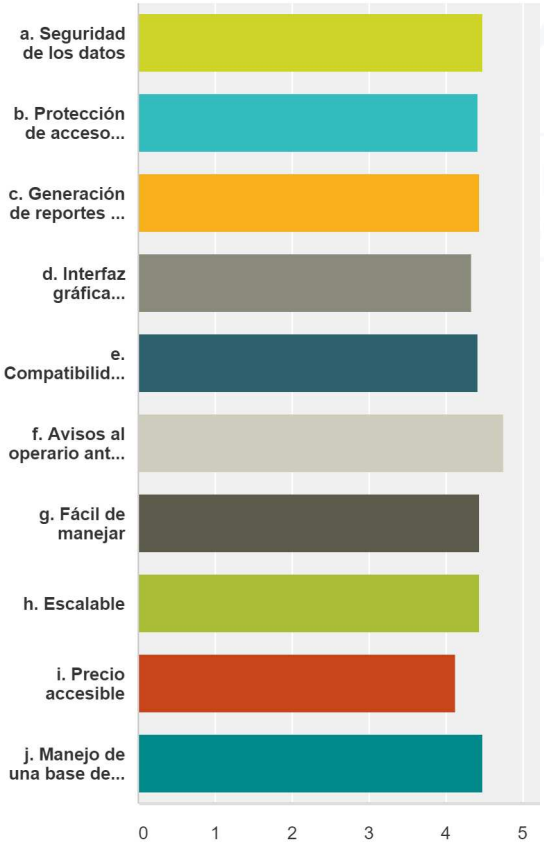
◯ En intervalos aleatorios de tiempo

GRACIAS POR SU COLABORACIÓN

Apéndice E

Resultados de las encuestas

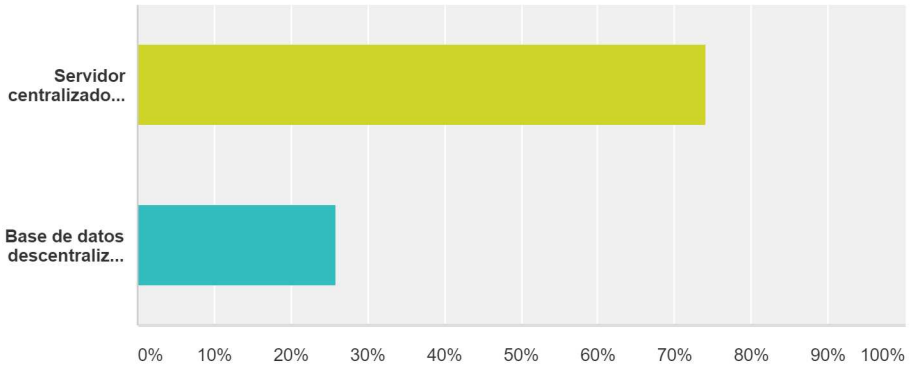
1. ¿Cuáles considera usted que son los requerimientos más importantes en un sistema SCADA industrial? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)



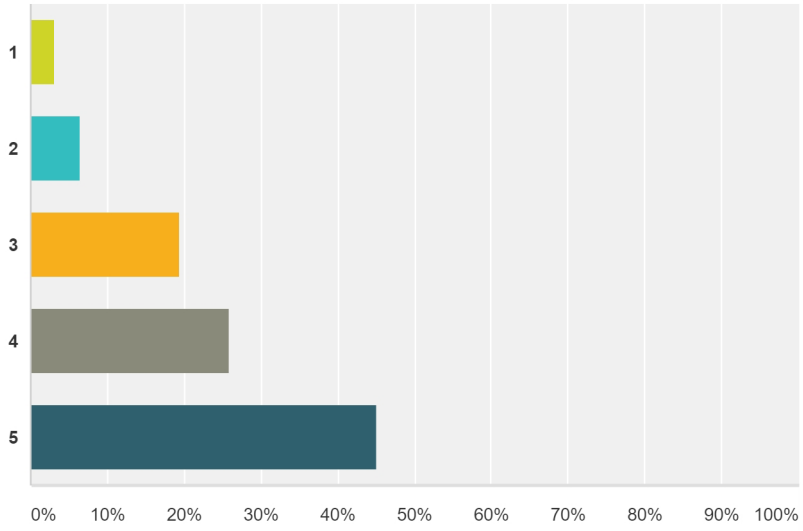
k.Otros:

- Accesibilidad remota.
- Flexible, para adecuarse a cualquier industria.
- Posibilidad de acceso inalámbrico, número de tags razonables, sumarización y especificidad en las alarmas, protección de pantallas cuando el operador no esta usando el HMI para alargar la vida útil de los mismos.
- Flexibilidad.

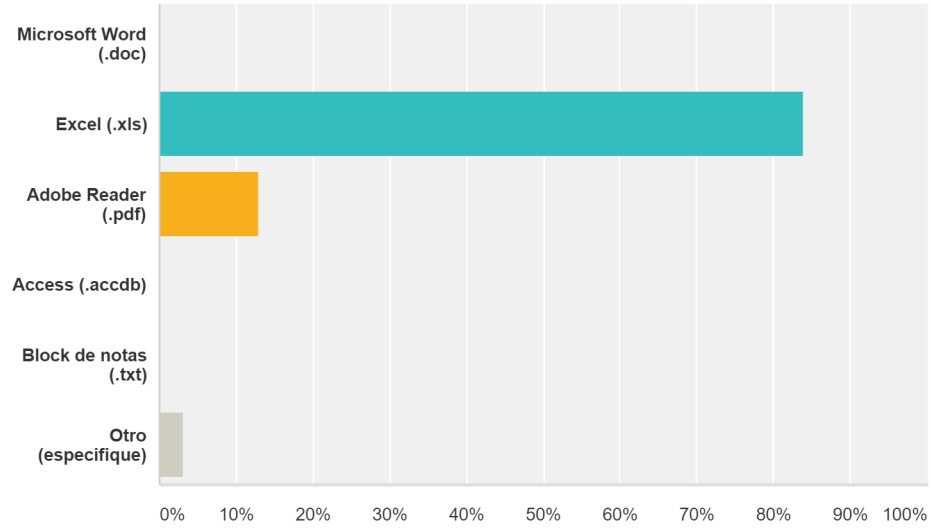
2. Cual considera usted que es la mejor arquitectura para la base de datos de un sistema SCADA



3. Según su experiencia, ¿Cree usted importante que los sistemas SCADA sean multiplaforma, es decir funcionen en varios sistemas operativos? (Asigne un numero número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)



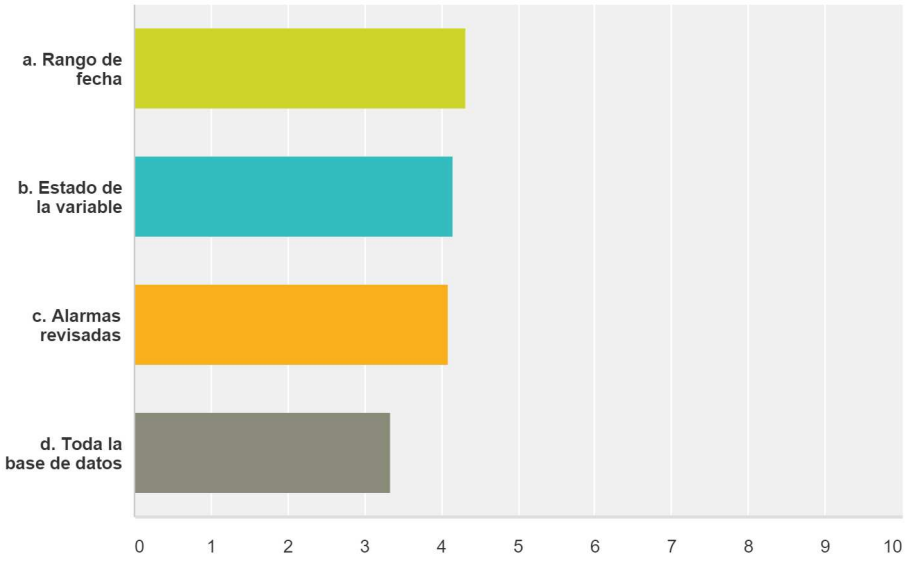
4. En que aplicación considera usted que se deben exportan los datos para generar un reporte.



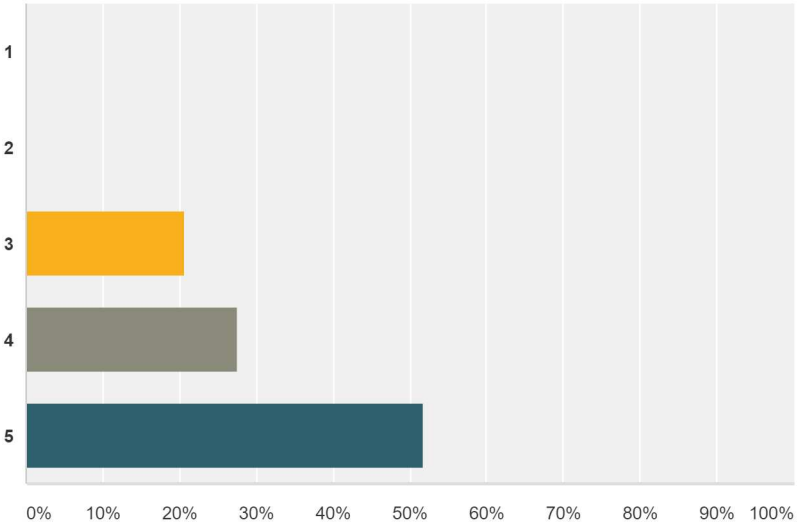
Otros:

- Debería ser en varias plataformas.

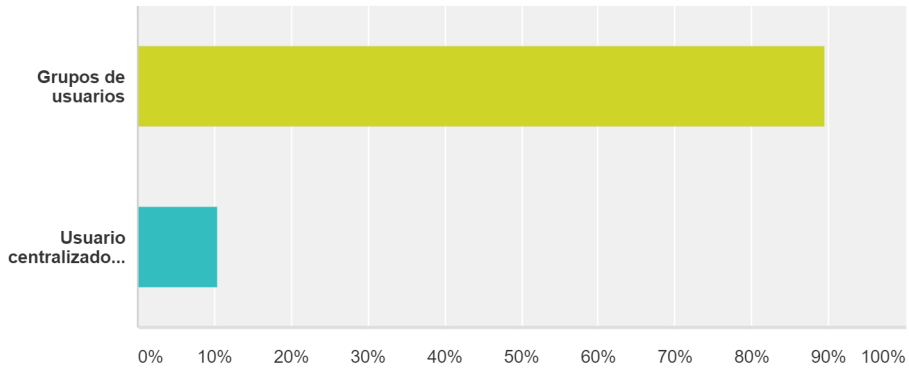
5. ¿Cuál considera usted que son las mejores búsquedas, dentro de la base de datos, para generar un reporte? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)



6. ¿Qué tan importante considera usted generar gráficas de tendencias de una variable? (Asigne un número de 1 al 5, siendo 5 lo más alto y 1 lo más bajo)

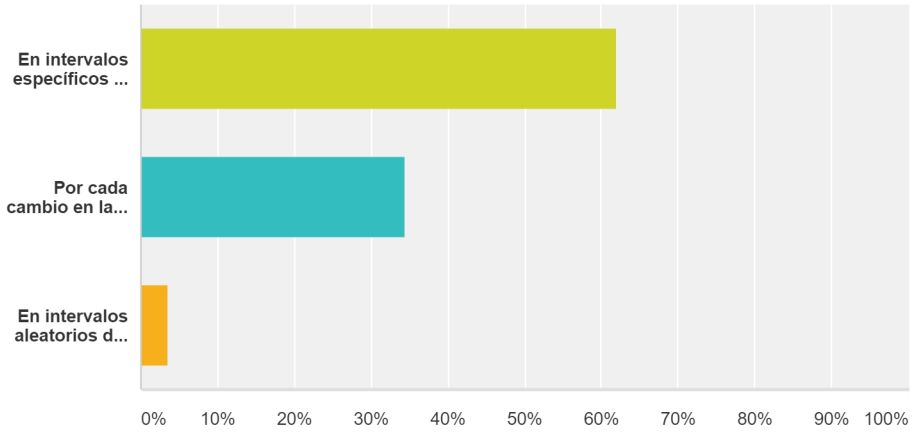


7. ¿Cuál considera usted que es la mejor administración de usuarios y protección de acceso?



8. ¿Cuál considera usted que sería el tiempo óptimo de respuesta en segundos del sistema al actualizar el estado de las variables?
Entre 3 y 5 segundos.

9. ¿Cuál considera usted que es la mejor manera de almacenar los datos en el servidor de base de datos?



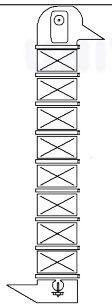
GRACIAS POR SU COLABORACIÓN

Apéndice F

Simbología de las interfaces SCADA

La Tabla F.1 muestra la simbología utilizada en las interfaces SCADA del sistema.

Tabla F.1: Tabla de simbología

SIMBOLOGÍA	DESCRIPCIÓN
	Alarma apagada
	Alarma encendida
	Motor apagada
	Motor encendido
	Banda transportadora
	Criba vibratoria
	Dosificación aditivo
	Dosificación clinker
	Dosificación puzolana seca
	Dosificación yeso
	Elevador cangilones
	Separador
	Silo almacenamiento
	Triturador

Apéndice G

Despliegue de la función de calidad

MATRIZ DE PLANEACIÓN

Requerimientos del Cliente / Requerimientos Técnicos	Importancia	Gestor de base de datos integrado	Actualización de los datos mediante peticiones	Encriptación de los datos	Autenticación de usuarios	Inserción de imágenes que representan los procesos	Métodos de busqueda dentro de la base de datos	Almacenamientoen la base de datos por cada cambio en la(s) variable(s) que se monitorea(n)	Envío de datos mediante sockets	Reporte al usuario del estado de los procesos mediante alarmas	Generación de reportes e históricos	Sistema basado en código libre	Facil de instalar y usar	Diseños Propuesto:	Competidor A: WinCC	Meta Planeada	Factor de Mejora	Carga Total	Porcentaje del Total
Seguridad de los datos	4		□	□	●				△					3	5	4	1,2	4,8	7%
Protección de acceso de cada usuario mediante autenticación	5				●								△	5	5	5	1	5	7%
Interfaz gráfica animada	3					□								3	4	4	1,2	3,6	5%
Compatibilidad con softwares de otras marcas	3								□			△		4	4	4	1	3	4%
Avisos al operario ante una falla en el sistema	5									●	△			5	5	5	1	5	7%
Fácil de manejar	4					△				△	△		●	4	4	4	1	4	6%
Escalabilidad de software y hardware	3								□			□		4	5	5	1,2	3,6	5%
Precio accesible	4											●		5	3	5	1	4	6%
Manejo de una base de datos	5	●									△			4	5	5	1,2	6	9%
Servidor centralizado para todos los dispositivos	4	△												2	5	4	1,4	5,6	8%
Funcione en varios sistemas operativos	4								△			●		4	1	4	1	4	6%
Exportar los datos en Excel	4										●			5	5	5	1	4	6%
Almacenamiento de los datos en intervalos específicos de tiempo	5		△					△						3	5	4	1,2	6	9%
Diferentes búsquedas dentro de la base de datos	3						□							4	4	5	1,2	3,6	5%
Actualización y monitoreo de los datos en intervalos específicos de tiempo	4		△					△						3	5	4	1,2	4,8	7%
Fácil de instalar	1												●	5	4	5	1	1	1%
PRIORIDAD TÉCNICA		59,6	24	14,4	88,2	14,8	10,8	10,8	28,6	49	51	85,8	50						
PRODUCTO B: WinCC	**Benchmarking**	1 para todo el sistema	cada segundo	Si	n usuarios	n en cada interfaz	n búsquedas	modificable	No	n alarmas por interfaz de usuario	n por cada interfaz de usuario	No	Si						
METAS DE DISEÑO		1 por cada PLC	cada 3 segundos	Si	n usuarios	n en cada interfaz	4 búsquedas	si	cualquier software que implemente TCP/IP	n alarmas por interfaz de usuario	n por cada interfaz de usuario	Si	Si						

Bibliografía

[1] M. Younus, C. Peiyong, L. Hu, and F. Yuqing, "Mes development and significant applications in manufacturing-a review," in *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, IEEE, 2010.

[2] C. Şahin and E. D. Bolat, "Development of remote control and monitoring of web-based distributed opc system," *Computer Standards & Interfaces*, 2009.

[3] J. E. S. Pinzon, "Diseño e implementación de sistemas scada para automatismos, basados en hardware y software libre," 2015.

[4] S. César San Martín, F. Torres, R. Barrientos, and M. Sandoval, "Monitoreo y control de temperatura de un estanque de agua entre chile y españa usando redes de alta velocidad," *REVISTA FACULTAD DE INGENIERÍA, UTA (CHILE)*, 2003.

[5] M. F. Bustos Castillo, "Diseño e implementación del sistema scada wincc de siemens a una máquina prototipo empacadora de galletas en el laboratorio de automatización de procesos de la upb," 2015.

[6] C. Davis, J. Tate, H. Okhravi, C. Grier, T. Overbye, and D. Nicol, "Scada cyber security testbed development," in *Proceedings of the 38th North American power symposium (NAPS 2006)*, 2006.

[7] C. Camargo, L. K. Duran, and N. F. Rosas, "Plataforma hardware/software abierta para aplicaciones en procesos de automatización industrial," *Ingenium*, 2013.

[8] V. Tipsuwanporn, A. Numsomran, S. Samaimak, and S. Harnnarong, "Development of redundant bus library for arduino to apply in scada system," in

Control, Automation and Systems (ICCAS), 2014 14th International Conference on, IEEE, 2014.

[9] D. V. Dutra, F. C. Posser, C. B. Tischer, L. G. Scherer, C. M. Franchi, and R. F. De Camargo, "Supervisory and control system development applied to distributed generation using web tools," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, IEEE, 2012.

[10] C. A. Sánchez Gabriel, "Desarrollo de sistema scada para el control de caudal basado en linux," *Universidad, Ciencia y Tecnología*, 2007.

[11] T. Turc and A. Gligor, "Development of service oriented web-based scada application," *Scientific Bulletin of the"Petru MaiorÜniversity of Targu Mures*, 2011.

[12] G. M. Zambrano-Rey, C. E. Fúquene-Retamoso, and H. S. Aguirre-Mayorga, "Aplicativo para el control estadístico de procesos en línea integrado a un sistema de manufactura flexible," *Ingeniería y universidad*, 2010.

[13] C. D. Acevedo Lara and R. A. Rueda Blanco, "Implementación de labview como sistema scada para la arquitectura de control snac pac opto 22, mediante una aplicación opc," 2013.

[14] J. Balcells, J. L. Romeral, and J. L. R. Martínez, *Autómatas programables*, vol. 1089. Marcombo, 1997.

[15] A. G. Higuera and F. J. C. García, *CIM, el computador en la automatización de la producción*, vol. 50. Univ de Castilla La Mancha, 2007.

[16] T. Szczepanski and T. Hadlich, "Opc-making the fieldbus interface transparent," tech. rep., Technical Report, OPC Foundation, 2003.

[17] A. R. Penin, *Sistemas Scada*. Marcombo, 2011.

[18] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[19] M. Schleipen, "Opc ua supporting the automated engineering of production monitoring and control systems," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, IEEE, 2008.

[20] J. Chunguo, W. Yan, and S. Xin, "Development of an opc server for remote monitoring and control based on gprs networks," in *Electronic Measurement & Instruments (ICEMI), 2011 10th International Conference on*, IEEE, 2011.

[21] M. S. Mahmoud, M. Sabih, and M. Elshafei, "Using opc technology to support the study of advanced process control," *ISA transactions*, 2015.

[22] N. C. System, *Supervisory Control and Data Acquisition (SCADA) Systems*. Arlington, Virginia: Technical Information Bulletin NCS TIB 04-1, 2004.

[23] J. Calderón, "Control y monitoreo scada de un proceso experimental, utilizando plc siemens s7-300 y software labview," Master's thesis, Universidad Nacional autónoma de México, México D.F, 2009.

[24] D. H. M. Grajales, Y. O. Sánchez, and M. Pinzón, "La confiabilidad, la disponibilidad y la mantenibilidad, disciplinas modernas aplicadas al mantenimiento," *Scientia et Technica*, 2006.

[25] M. R. Sol and R. M. Ortiz, *Diseño e implantación de un sistema SCADA para una planta de producción y envasado de líquidos*. PhD thesis, Universitat Autònoma de Barcelona, 2008.

[26] M. Reinoso, "Reporte de los diagramas p&id de instrumentación del área de molienda de cemento de ucem planta industrial guapán," *Industria Guapán*, 2015.

[27] D. Nardella, *Snap7 Reference manual*, Enero 2014.

[28] M. J. Donahoo and K. L. Calvert, *TCP/IP sockets in C: practical guide for programmers*. Morgan Kaufmann, 2009.

[29] J. Stepisnik, *Distributed Object-Oriented Architectures*. Diplomica GmbH, 2007.

[30] T. G. Pere Ponsa, "Diseño industrial, diseño de pantalla gedis," *Universidad Politécnica de Cataluña (UPC)*.

[31] S. AG, "Simatic wincc flexible," *Siemens*, 2009.

[32] E. Yacuzzi and F. Martín, "Qfd: Conceptos, aplicaciones y nuevos desarrollos," tech. rep., Serie Documentos de Trabajo, Universidad del CEMA: Área: negocios, 2003.